DRES
94-0681
12321

DRES

DEFENCE RESEARCH ESTABLISHMENT SUFFIELD

**CR 94-29**

***UNCLASSIFIED***

# INVESTIGATION OF SIMULATED FLIGHT CONTROL USING AN INVERSE DYNAMIC MODEL NEURAL NETWORK (U)

BY

ATLANTIS AEROSPACE CORPORATION

SCIENTIFIC AUTHORITY: DR. S. BARTON

AUGUST 1994

**WARNING**
"The use of this information is permitted subject to recognition of proprietary and patent rights."

CRAD

National Defence Défence nationale Canada

*Atlantis Aerospace Corporation*
*Brampton, Ontario*
*Canada*

*Document Number: ED997-00138*
*Revision: 1.01*
*Issue Date: 28 April 1994*

# INVESTIGATION OF SIMULATED FLIGHT CONTROL USING AN INVERSE DYNAMIC MODEL NEURAL NETWORK

## FINAL REPORT

Contract No.: W7702-2-R320/01-XSG

**Prepared For:**

**Advanced Guidance Concepts Group / Systems Integration Group**
**Defense Research Establishment Suffield**
**Medicine Hat, Alberta**
**Canada**

**Approvals**

| Title | Name | Signature |
|---|---|---|
| Programme Manager | Caroline Cheung | |
| Author | Mark Langley | |
| Department Manager | Eric Solowka | |
| Technical Publications QA | N/A | |

## ABSTRACT

This document is the final report prepared in fulfilment of the DND R&D Contract DSS File No. W7702-2-R320. It summarizes the activities completed by Atlantis Aerospace Corporation during Phases II, III, and IV of this contract, as part of DRES' research and development of unmanned vehicle guidance and control technologies. In particular, the work investigated the application of neural networks and innovative control techniques to guidance of an unmanned air vehicle.

## ACKNOWLEDGEMENT

Significant aspects of the work described in this R&D contract required Atlantis to investigate relatively new and potentially promising technologies. Atlantis is highly appreciative of the opportunity to participate and contribute to the efforts to meet the Canadian Forces autonomous vehicle requirements. The scientific and technical leadership of the Advanced Guidance Concepts Group is particularly appreciated.

## TABLE OF CONTENTS

**Section** **Page**

1.0 Introduction . . . . . . . . . . 1

2.0 Referenced Documents . . . . . . . . . . 1

3.0 Genetic Algorithm Evaluation . . . . . . . . . . 2
  3.1 Introduction . . . . . . . . . . 2
  3.2 Dynamic Model . . . . . . . . . . 2
  3.3 Controller Design . . . . . . . . . . 3
  3.4 Evaluation Function . . . . . . . . . . 4
  3.5 Training Results . . . . . . . . . . 5

4.0 FLISIM Integration . . . . . . . . . . 11
  4.1 Introduction . . . . . . . . . . 11
  4.2 FLISIM Adaptation to the i860 environment . . . . . . . . . . 11
  4.3 Simplifying Assumptions . . . . . . . . . . 12

5.0 Longitudinal Neural Network Development . . . . . . . . . . 14
  5.1 Training Data Generation . . . . . . . . . . 14
  5.2 Network Training Results . . . . . . . . . . 15
  5.3 Network Performance . . . . . . . . . . 17
  5.4 Online Training Results . . . . . . . . . . 22
    5.4.1 IDM Error Network Architecture . . . . . . . . . . 24
    5.4.2 Inverse/Forward Dynamic Model . . . . . . . . . . 25
    5.4.3 IDM Network with Proportional Training . . . . . . . . . . 27

6.0 Application to Off-Nominal Vehicle Configurations . . . . . . . . . . 31
  6.1 Off-Nominal Case 1 - Servo Positioning Offset . . . . . . . . . . 31
  6.2 Off-Nominal Case 2 - Reduced Canard Effectiveness . . . . . . . . . . 33
  6.3 Off-Nominal Case 3 - Increased Mass and Inertia . . . . . . . . . . 33

7.0 Application of Genetic Algorithms to Longitudinal Controller Optimization 36
  7.1 Introduction . . . . . . . . . . 36
  7.2 Genetic Algorithm Implementation . . . . . . . . . . 36
  7.3 Genetic Algorithm Results . . . . . . . . . . 37
  7.4 Conclusions . . . . . . . . . . 42

8.0 Lateral/Longitudinal Neural Network Development . . . . . . . . . . 44
  8.1 Training Data Generation . . . . . . . . . . 44
  8.2 IDM Network Training . . . . . . . . . . 44
  8.3 Autopilot Integration . . . . . . . . . . 46
  8.4 Test Flight Results . . . . . . . . . . 47

8.4.1 Lateral Test Flight 1 . . . . . 48
8.4.2 Lateral Test Flight 2 . . . . . 51
8.4.3 Lateral Test Flight 3 . . . . . 54
8.5 Network Stability Analysis . . . . . 57

9.0 Application to Off-Nominal Conditions . . . . . 63
9.1 Controller Sensitivity to Sensor Noise . . . . . 63
9.2 Increased Servo Lag . . . . . 66
9.3 Constant Servo Error . . . . . 69

10.0 Application of Genetic Algorithms to Lateral/Longitudinal Controller Optimization . . . . . 72
10.1 Test Configuration . . . . . 72
10.2 . . . . . 74

11.0 Real Flight Requirements Definition . . . . . 78
11.1 Algorithmic and Computational Requirements . . . . . 78
11.2 Neural Network Hardware . . . . . 78
11.3 Communication Interface . . . . . 80
11.4 Areas of Potential Risk . . . . . 81
11.4.1 Observability of Required Inputs . . . . . 81
11.4.2 Accuracy of Training Data . . . . . 81
11.4.3 Sensitivity to Sensor Noise . . . . . 81
11.4.4 Sanity Checks and Override Procedures . . . . . 82
11.5 Flight Test Plan . . . . . 82

12.0 Recommendations for Future Work . . . . . 83

13.0 List of Deliverables . . . . . 85

14.0 References . . . . . 86

## LIST OF ILLUSTRATIONS

**Figure Number/Title** **Page**

Figure 3.1: Simulated tractor trailer ... 2
Figure 3.2: GA Convergence for the Truck Backer-Upper ... 6
Figure 3.3: GA Convergence for the Truck Backer-Upper (expanded view) ... 6
Figure 3.4: Truck course (initial state 1) ... 8
Figure 3.5: Truck course (initial state 2) ... 8
Figure 3.6: Truck course (initial state 3) ... 8
Figure 3.7: Truck course (initial state 4) ... 8
Figure 3.8: Truck course (initial state 5) ... 8
Figure 3.9: Truck course (initial state 6) ... 8
Figure 3.10: Truck course (initial state 7) ... 9
Figure 3.11: Truck course (initial state 8) ... 9
Figure 3.12: Truck course (initial state 9) ... 9
Figure 3.13: Truck course (initial state 10) ... 9
Figure 3.14: Truck course (initial state 11) ... 9
Figure 3.15: Truck course (initial state 12) ... 9
Figure 3.16: Truck course (new initial state 1) ... 10
Figure 3.17: Truck course (new initial state 2) ... 10
Figure 3.18: Truck course (new initial state 3) ... 10
Figure 3.19: Truck course (new initial state 4) ... 10
Figure 3.20: Truck course (new initial state 5) ... 10
Figure 3.21: Truck course (new initial state 6) ... 10
Figure 5.1: IDM Network Offline Convergence ... 16
Figure 5.2: MPS-100 Altitude Control Law (PID version) ... 17
Figure 5.3: MPS-100 Altitude Control Law (Neural Network Variant) ... 18
Figure 5.4: PID and NN Controller Altitude Profiles - Case 1 ... 20
Figure 5.5: PID and NN Controller Canard Profiles - Case 1 ... 20
Figure 5.6: PID and NN Controller Altitude Profiles - Case 2 ... 21
Figure 5.7: PID and NN Controller Canard Profiles - Case 2 ... 21
Figure 5.8: IDM network online training results ... 22
Figure 5.9: Inverse Dynamic Network with Error Estimation ... 24
Figure 5.10: Effect of error network on online training performance ... 25
Figure 5.11: IDM/FDM Controller Architecture ... 26
Figure 5.12: Effect of IDM/FDM Architecture on Online Training Performance . 27
Figure 5.13: Altitude Profile - Proportional Learning ... 30
Figure 5.14: Flight Path Angle Errors - Proportional Training ... 30
Figure 6.1: Altitude Profile - 2° Canard Offset ... 32
Figure 6.2: Flight Path Angle Errors - 2° Canard Offset ... 32
Figure 6.3: Altitude Profile - Reduced Canard Effectiveness ... 34
Figure 6.4: Flight Path Angle Errors - Reduced Canard Effectiveness ... 34
Figure 6.5: Altitude Profile - Increased Mass and Inertia ... 35
Figure 6.6: Flight Path Angle Errors - Increased Mass and Inertia ... 35

Figure 7.1: Average Population Scores . . . . . . . . . . . . . . . . . . . . . . . . . . 38
Figure 7.2: Best of Generation Scores . . . . . . . . . . . . . . . . . . . . . . . . . . . 38
Figure 7.3: Genetically Optimized Controller - Training Flight Profile . . . . . . . . . 41
Figure 7.4: Genetically Optimized Controller - Novel Flight Profile . . . . . . . . . 41
Figure 8.1: PID Controller Lateral Control Loop . . . . . . . . . . . . . . . . . . . . . . 46
Figure 8.2: NN Controller Lateral Control Loop . . . . . . . . . . . . . . . . . . . . . . 46
Figure 8.3: Aileron History - Lateral Test Flight 1 . . . . . . . . . . . . . . . . . . . . 48
Figure 8.4: Canard History - Lateral Test Flight 1 . . . . . . . . . . . . . . . . . . . . 49
Figure 8.5: Roll Angle History - Lateral Test Flight 1 . . . . . . . . . . . . . . . . . . 49
Figure 8.6: Heading Angle History - Lateral Test Flight 1 . . . . . . . . . . . . . . . 50
Figure 8.7: Altitude History - Lateral Test Flight 1 . . . . . . . . . . . . . . . . . . . 50
Figure 8.8: Aileron History - Lateral Test Flight 2 . . . . . . . . . . . . . . . . . . . . 51
Figure 8.9: Canard History - Lateral Test Flight 2 . . . . . . . . . . . . . . . . . . . . 52
Figure 8.10: Roll Angle History - Lateral Test Flight 2 . . . . . . . . . . . . . . . . . 52
Figure 8.11: Heading Angle History - Lateral Test Flight 2 . . . . . . . . . . . . . . 53
Figure 8.12: Altitude History - Lateral Test Flight 2 . . . . . . . . . . . . . . . . . . 53
Figure 8.13: Aileron History - Lateral Test Flight 3 . . . . . . . . . . . . . . . . . . . 54
Figure 8.14: Canard History - Lateral Test Flight 3 . . . . . . . . . . . . . . . . . . . 55
Figure 8.15: Roll Angle History - Lateral Test Flight 3 . . . . . . . . . . . . . . . . . 55
Figure 8.16: Heading Angle History - Lateral Test Flight 3 . . . . . . . . . . . . . . 56
Figure 8.17: Altitude History - Lateral Test Flight 3 . . . . . . . . . . . . . . . . . . 56
Figure 8.18: Aileron History - Modified Network Inputs . . . . . . . . . . . . . . . . 60
Figure 8.19: Canard History - Modified Network Inputs . . . . . . . . . . . . . . . . 60
Figure 8.20: Roll Angle History - Modified Network Inputs . . . . . . . . . . . . . . 61
Figure 8.21: Heading Angle History - Modified Network Inputs . . . . . . . . . . . . 61
Figure 8.22: Altitude History - Modified Network Inputs . . . . . . . . . . . . . . . . 62
Figure 9.1: Altitude vs. Time - 3°/s Rate Gyro Noise . . . . . . . . . . . . . . . . . . 64
Figure 9.2: Roll Angle vs. Time - 3°/s Rate Gyro Noise . . . . . . . . . . . . . . . . 64
Figure 9.3: Altitude vs. Time - 10°/s Rate Gyro Noise . . . . . . . . . . . . . . . . . 65
Figure 9.4: Roll Angle vs. Time - 10°/s Rate Gyro Noise . . . . . . . . . . . . . . . 65
Figure 9.5: Altitude vs. Time - Increased Servo Lag . . . . . . . . . . . . . . . . . . 66
Figure 9.6: Roll Angle vs. Time - Increased Servo Lag, PID Controller . . . . . . 67
Figure 9.7: Roll Angle vs. Time - Increased Servo Lag, NN Controller . . . . . . . 67
Figure 9.8: Altitude vs. Time - Constant Servo Error . . . . . . . . . . . . . . . . . . 69
Figure 9.9: Roll Angle vs. Time - Constant Servo Error, PID Controller . . . . . . 70
Figure 9.10: Roll Angle vs. Time - Constant Servo Error, NN Controller . . . . . 70
Figure 10.1: Best of Generation Scores for Longitudinal/Lateral Controller Development . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 73
Figure 10.2: Population Evaluation Profiles - Selected Generations . . . . . . . . . 74
Figure 10.3: Genetically Optimized Altitude Profile . . . . . . . . . . . . . . . . . . . 75
Figure 10.4: Genetically Optimized Heading Profile . . . . . . . . . . . . . . . . . . . 76
Figure 10.5: 486-50 Process Network Throughput . . . . . . . . . . . . . . . . . . . 79

# LIST OF TABLES

**Table Number/Title** **Page**

Table 3.1: Initial states used for truck controller evaluation . . . . . . . . . . . . . 5
Table 5.1: IDM Network Input and Output Formats . . . . . . . . . . . . . . . . . . . 16
Table 5.2: IDM Network Convergence Results . . . . . . . . . . . . . . . . . . . . . . 17
Table 5.3: Commanded Altitude Profiles for Neural Network Evaluation Flights 18
Table 6.1: Off-Nominal Vehicle Configurations . . . . . . . . . . . . . . . . . . . . . . 31
Table 7.1: Longitudinal Control Law Parameters Altered by Genetic Algorithm 37
Table 7.2: Altitude Profile for Evaluation of Genetically Derived Controllers . . 37
Table 7.3: Gain Settings for GA-Controller Evaluations . . . . . . . . . . . . . . . . 40
Table 8.1: I/O Definition for Longitudinal/Lateral IDM Networks . . . . . . . . . . 45
Table 8.2: Mission Profile Settings for Lateral/Longitudinal Test Flights . . . . . 47
Table 10.1: Lateral Control Law Parameters Altered by Genetic Algorithm . . . 73

## 1.0 Introduction

Atlantis Aerospace possesses extensive experience in the design and integration of high performance autopilot systems and sensor suites, and has produced autopilot systems for a number of clients. Atlantis designed and implemented the microPilot autopilot system used for guidance and control of the ROBOT-X and HATT-X unmanned vehicles. Under contract to the Defense Research Establishment Suffield (DRES), Atlantis conducts continuing investigations into enhancing the capabilities of the HATT-X vehicle and its autopilot.

It is anticipated that future autopilots will derive increased capability and greater autonomy from the application of artificial intelligence techniques to vehicle guidance and control. This contract investigates the applicability of neural networks to synthesis of control software for a high performance aircraft autopilot. The work conducted has been administered by Supply and Services Canada under contract number W7702-2-R320/01-XSG, with technical authority provided by the Advanced Guidance Concepts Group, Defense Research Establishment Suffield.

This report covers the work completed by Atlantis Aerospace under the final phases of the above contract. Specifically, it documents Atlantis's work in developing a neural network controller for the HATT-X unmanned vehicle. Each stage of controller development is presented. Problems encountered, their solutions, and evaluation results are included for each development stage. Finally, necessary requirements and recommendations are presented to allow integration of a neural network controller with the actual HATT-X vehicle.

Work conducted by Atlantis in fulfilment of the Task 1 requirements of this project are not included in this report. This work is discussed in the companion document ED999-00242 Rev. 1.01 (Investigation of Simulated Flight Control Using An Inverse Dynamic Model Neural Network - Interim Report).

## 2.0 Referenced Documents

The following documents are referenced in this document:

| | |
|---|---|
| ED999-00242 Rev 1.01 | Investigation of Simulated Flight Control Using An Inverse Dynamic Model Neural Network - Interim Report |

## 3.0 Genetic Algorithm Evaluation

### 3.1 Introduction

As the final task of Phase I, genetic algorithms were used to develop an outer loop controller for the simple aircraft model. The resulting controller was shown to produce good performance across the set of input conditions. Accordingly, genetic algorithms were identified as a potentially promising approach to design of an outer loop controller for the HATT-X IDM network to be developed in Phase II.

To further evaluate its capabilities, Atlantis proposed to apply the genetic algorithm technique to the control of other simple dynamic systems. The selected problem was that of backing a tractor trailer up to a loading dock. This problem was selected for the three following reasons.

1. Backing up a tractor trailer to a loading dock is a widely used benchmark in the design of artificially intelligent controllers. Evaluation of results is facilitated by comparison with the existing body of research.
2. The dynamics of the tractor trailer are simply modelled, reducing the time required to produce a working simulation.
3. The tractor trailer dynamics are highly non-linear, thereby providing a good test of controller robustness.

### 3.2 Dynamic Model

The primary reference for this application was provided by the work of Kong and Kosko[1], in which a fuzzy logic truck controller was designed and evaluated. The original statement of the truck and trailer problem is presented in Nguyen and Widrow[2], in which a neural network truck controller was developed.

Each of these papers provides an overview of the problem domain without including a detailed description of the truck dynamics. An appropriate state description and dynamic equations were therefore formulated by Atlantis. The truck state is illustrated in Figure 3.1 and is defined by the equations

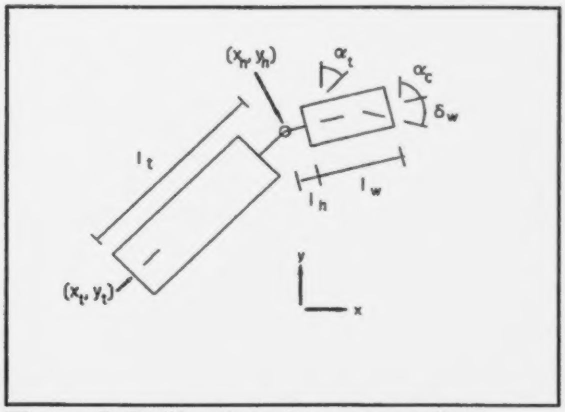

**Figure 3.1**: Simulated tractor trailer

$$\dot{\alpha}_c=\frac{\delta_w}{l_w} \qquad (3.1)$$

$$\dot{\alpha}_t=\frac{\sin(\alpha_c-\alpha_t)}{l_t} \qquad (3.2)$$

$$(3.3) \qquad \dot{x}_t=\cos(\alpha_c-\alpha_t)\sin(\alpha_t)$$

$$\dot{y}_t=\cos(\alpha_c-\alpha_t)\cos(\alpha_t) \qquad (3.4)$$

$$(\alpha_c)_{new}=\alpha_c+\dot{\alpha}_c\Delta t \qquad (3.5)$$

$$(\alpha_t)_{new}=\alpha_t+\dot{\alpha}_t\Delta t \qquad (3.6)$$

$$(x_t)_{new}=x_t+\dot{x}_t\Delta t \qquad (3.7)$$

$$(y_t)_{new}=y_t+\dot{y}_t\Delta t \qquad (3.8)$$

$$(x_h)_{new}=(x_t)_{new}+l_t\sin(\alpha_t) \qquad (3.9)$$

$$(y_h)_{new}=(y_t)_{new}+l_t\cos(\alpha_t) \qquad (3.10)$$

where $\alpha_c$ and $\alpha_t$ are the angles between the vertical and the cab and trailer respectively. $x_t$ and $y_t$ define the location of the trailer rear. $l_t$, $l_h$, and $l_w$ are the trailer length, trailer hitch length, and truck wheelbase respectively, while $\delta_w$ is the commanded steering angle.

## 3.3 Controller Design

The design goal for this application was to produce a traditional linear control law which could successfully reverse the truck from each set of initial conditions. The intent of this research was to further evaluate the optimization capabilities of the genetic algorithm. Accordingly, the genetically derived controller was applied directly to the plant. No benefit was seen to using an inverse dynamic network, and the time required to train and validate the network would have been significant.

The controller input consisted of the six variables $x_t$, $y_t$, $\sin(\alpha_c)$, $\cos(\alpha_c)$, $\sin(\alpha_t)$, and $\cos(\alpha_t)$. The generic control law was based on three general rules developed through manual attempts to reverse the truck. First, the angle between the cab and trailer must be kept small to prevent jack knifing. Second, the trailer should be pointed toward the gate. Finally, the trailer angle must be near ninety degrees when the gate is reached.

On the basis of these heuristics, the control law format attempted was given by the equations.

$$e_{cab}=\min(\max(\alpha_t-\alpha_c,g_0),g_1) \tag{3.11}$$

$$e_{trailer}=\min(\max(90-\alpha_t,g_2),g_3) \tag{3.12}$$

$$e_{heading}=\min\left\{\max\left[\tan^{-1}\left(\frac{x_t}{y_t}\right)-\alpha_t,g_4\right],g_5\right\}$$

$$\delta_w=g_6\times e_{cab}+g_7\times e_{trailer}+g_8\times e_{heading}$$

where $e_{cab}$ denotes the magnitude limited difference between the cab and trailer angles. $e_{trailer}$ is the angular error between the trailer angle and desired final angle. $e_{heading}$ is the discrepancy between trailer angle and the direction vector from the trailer rear to the loading dock. Finally, $g_0$ through $g_8$ define the nine-element parameter vector optimized by the genetic algorithm.

### 3.4 Evaluation Function

The fitness of each parameter vector generated by the genetic algorithms was evaluated according to the following criteria.

1. If $|\alpha_c - \alpha_t|$ is greater than 90°, the truck is jack knifed, and the evaluation assigned is - (1,000,000 - time elapsed before jack knifing). This rule has the effect of heavily penalizing maneuvers which result in jack knifing. By incorporating a time based term, an individual which survives longer before jack knifing is not punished as severely and is therefore more likely to reproduce.

2. If the rear of the truck leaves the yard area defined by $0 < x < 60$, $-25 < y < 25$, the evaluation assigned is given by

$$100 * \left[ \frac{1}{5} |\alpha_t - 90°| + |y_t| + |x_t| \right] \tag{3.15}$$

3. If any portion of the truck crosses the plane defined by $x = 0$, an evaluation equal to that in rule 2 is applied.

The total score for each controller was determined by reversing the tractor from a set of twelve initial geometries, as defined in Table 3.1. Rules 1 through 3 were applied to each of the resulting trajectories and the twelve scores produced were summed to yield the total evaluation for the individual.

**Table 3.1**: Initial states used for truck controller evaluation

| | Initial Condition # | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| $x_t$ (m) | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 |
| $y_t$ (m) | -5 | -5 | -5 | -5 | 0 | 0 | 0 | 0 | 5 | 5 | 5 | 5 |
| $\alpha_t$ (°) | -45 | 0 | 45 | 90 | -45 | 0 | 45 | 90 | -45 | 0 | 45 | 90 |
| $\alpha_e$ (°) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## 3.5 Training Results

The dynamic model and evaluation functions described in the preceding sections were input to the genetic algorithm and used to generate a vehicle controller. This procedure was repeated several times, beginning with a randomly generated population for each run. In all cases, convergence occurred within 24 to 72 hours of 486 CPU time. An example time history of controller performance is depicted in Figure 3.2 and Figure 3.3.

The vehicle trajectories resulting from the genetically designed controller are shown in Figure 3.4 through Figure 3.15 for the twelve initial geometries used in controller design. The ability of the controller to generalize to unknown initial configurations is depicted in Figure 3.16 through Figure 3.21.

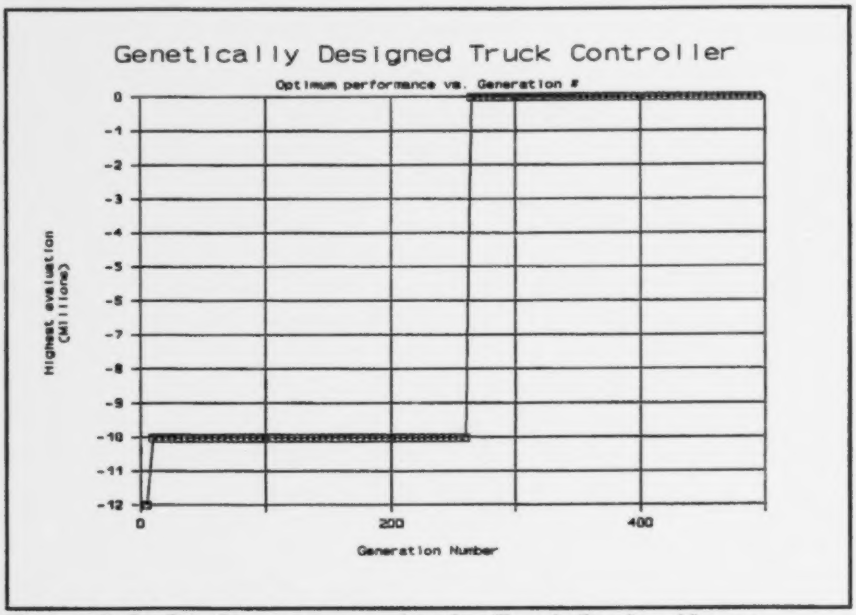

**Figure 3.2:** GA Convergence for the Truck Backer-Upper

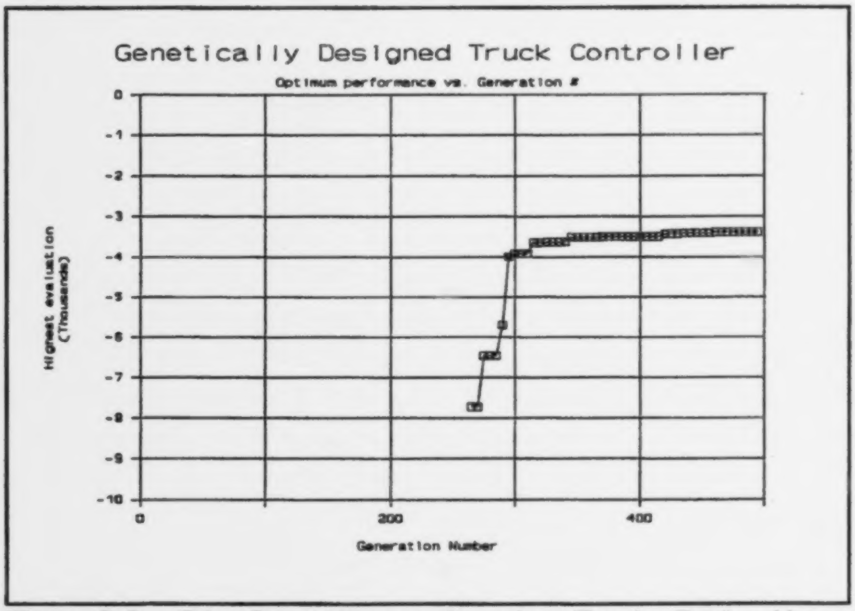

**Figure 3.3:** GA Convergence for the Truck Backer-Upper (expanded view)

Examination of Figure 3.4 through Figure 3.15 indicates that the genetically designed controller provides excellent control when executed over the trajectories used in training. This is an expected result, as the selected gain set is the "optimum" for these trajectories, where "optimum" is a subjective quantity chosen by the designer and described by the selected performance equation.

The controller performance is degraded in the runs of Figure 3.16 through Figure 3.21 where it is required to generalize to novel initial conditions. For comparative purposes, an option for manually reversing the truck was added to the simulation software. Human operators were allowed several practice runs to gain familiarization with the truck dynamics. Despite this allowance, the test subjects' performances were generally inferior to those of the genetically designed controller on both the training and generalization initial conditions. The sole exception to this rule is found in the initial conditions of Figure 3.19. For this case, the human operators typically produced an improved final truck orientation.

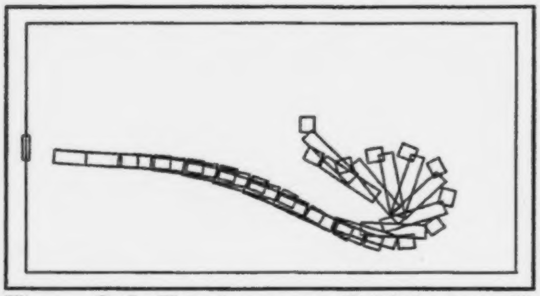

**Figure 3.4:** Truck course (initial state 1)

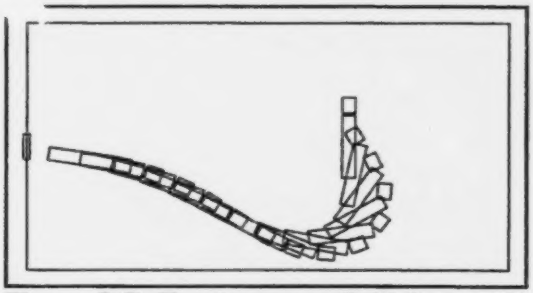

**Figure 3.5:** Truck course (initial state 2)

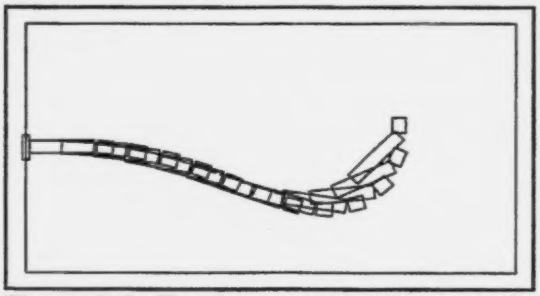

**Figure 3.6:** Truck course (initial state 3)

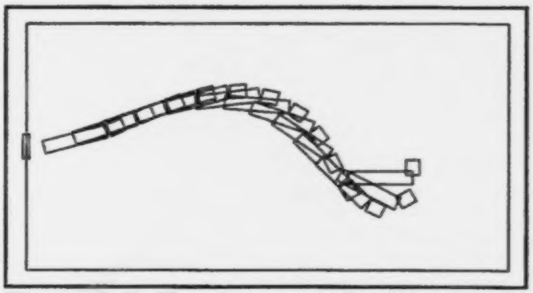

**Figure 3.7:** Truck course (initial state 4)

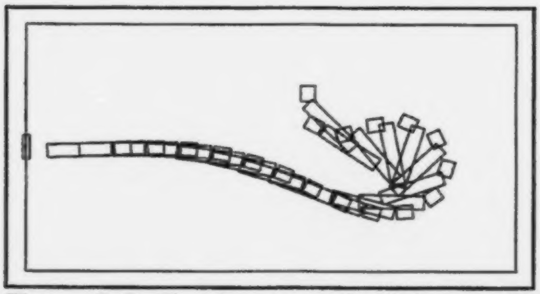

**Figure 3.8:** Truck course (initial state 5)

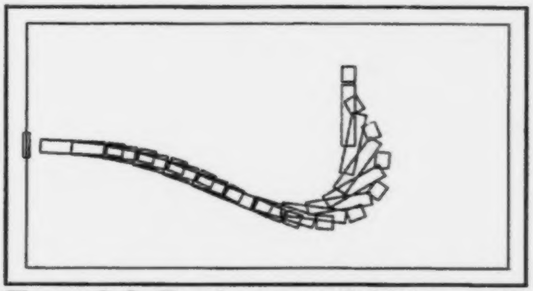

**Figure 3.9:** Truck course (initial state 6)

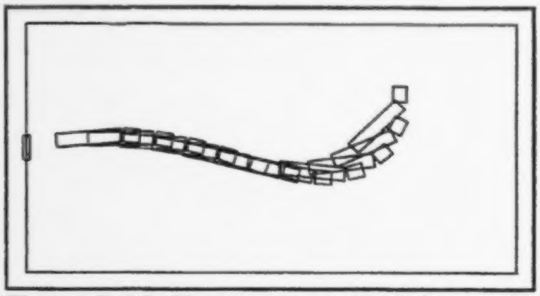

**Figure 3.10:** Truck course (initial state 7)

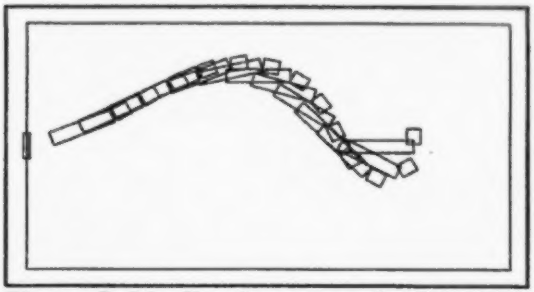

**Figure 3.11:** Truck course (initial state 8)

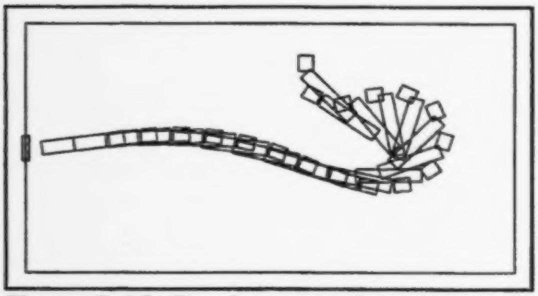

**Figure 3.12:** Truck course (initial state 9)

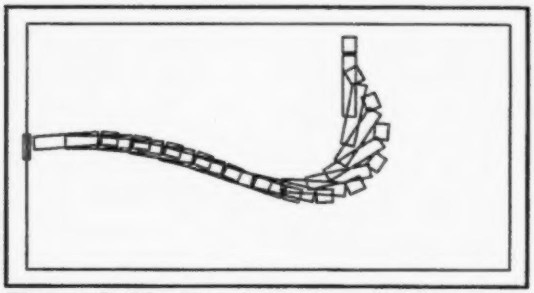

**Figure 3.13:** Truck course (initial state 10)

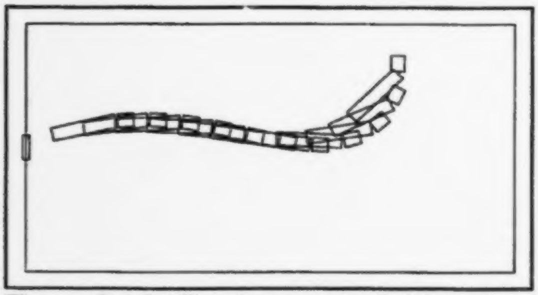

**Figure 3.14:** Truck course (initial state 11)

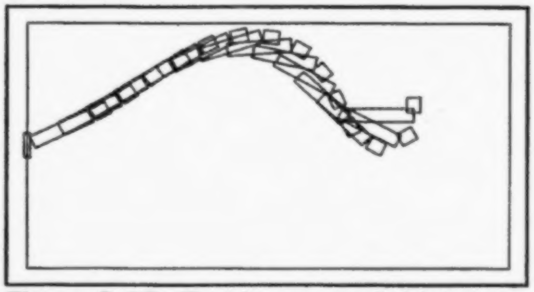

**Figure 3.15:** Truck course (initial state 12)

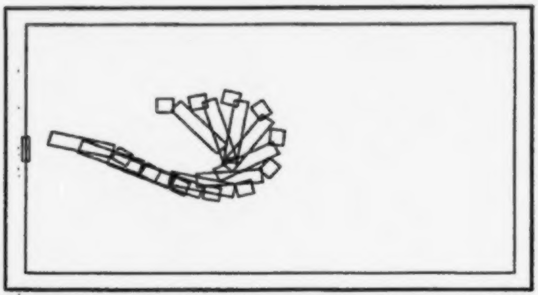

**Figure 3.16:** Truck course (new initial state 1)

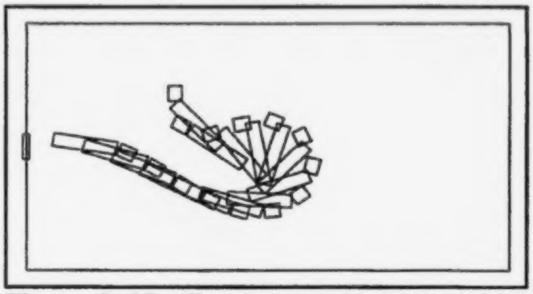

**Figure 3.17:** Truck course (new initial state 2)

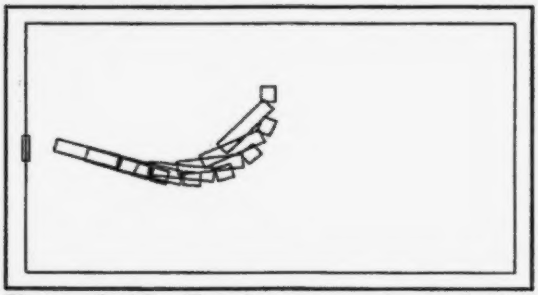

**Figure 3.18:** Truck course (new initial state 3)

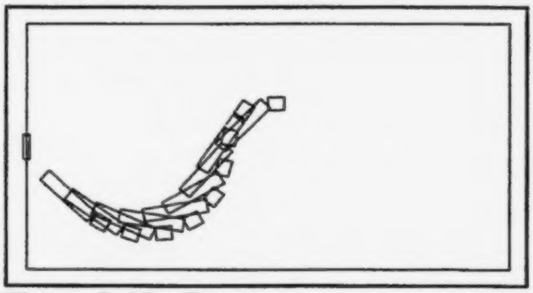

**Figure 3.19:** Truck course (new initial state 4)

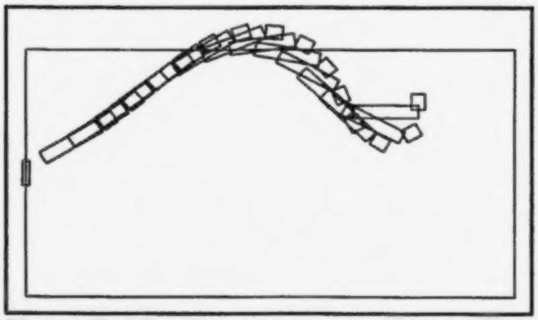

**Figure 3.20:** Truck course (new initial state 5)

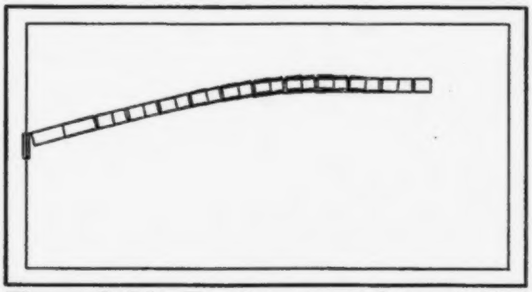

**Figure 3.21:** Truck course (new initial state 6)

## 4.0 FLISIM Integration

### 4.1 Introduction

In fulfillment of the contractual requirements for Task II, Atlantis investigated the application of the inverse dynamic model neural network (IDM NN) to pitch control of the HATT-X research vehicle. This work was conducted using the FLISIM flight simulation software package developed by DRES. Addition of a neural network capability to the FLISIM package proceeded in several stages:

a. adaptation and validation of the UNIX-based FLISIM software for use on the NumberSmasher 860 PC,
b. incorporation of simplifying assumptions particular to the neural network research program,
c. generation of training data files,
d. off-line neural network training,
e. on-line network validation and performance comparison against the existing autopilot simulation.
f. on-line generation of optimal control laws using genetic algorithms.

Each development phase is discussed in detail in the following sections.

### 4.2 FLISIM Adaptation to the i860 environment

Under the original proposal submitted by Atlantis, integration of neural networks with FLISIM was to be conducted on-site at DRES, using the existing FLISIM software. Atlantis subsequently realized that substantial savings could be incurred by porting the FLISIM software for use on the NumberSmasher 860 PC procured under Task I of the contract. This approach was approved by the Scientific Authority on the basis of the following arguments:

a. travel and housing expenses associated with on-site work at DRES were eliminated, as all development and operation was conducted at Atlantis.

b. NumberSmasher 860 PC computational speed exceeds that of the DECstations used at DRES.

c. Existing neural network software developed and validated by Atlantis under Task I can be directly interfaced to the FLISIM software on the NumberSmasher PC. Implementation on the DRES DECstations would have required a rewrite and translation of the neural network software from "C" to Fortran.

Accordingly, Atlantis completed a port of the FLISIM software to the NumberSmasher PC. This work proceeded smoothly, with few modifications required. The remainder of this section itemizes the changes which were necessary. As these will not be of general interest, the reader may skip over this section without loss of content.

All changes implemented were a result of the varying Fortran standards in existence. In particular, the Microway compiler used on the NumberSmasher PC implements the Fortran 77 standard, while the FLISIM software assumes the availability of VMS and Microsoft extensions to the Fortran language. As a result, the following changes were required:

a. Specification of the ASSOCIATEVARIABLE= clause in file OPEN() statements is ignored by the Microway compiler. The FLISIM code was therefore modified to manually update the associated index variable after each file read or write.

b. Microway Fortran conforms to the Fortran 77 standard of expressing file record lengths as a total number of bytes, while the FLISIM code assumes the VMS method of specifying lengths as an equivalent number of long words. To maintain consistency, record lengths in the PC version of the code were multiplied by four when opening fixed length files.

c. Data formats for direct access files differ between the DECstation and the NumberSmasher PC. New direct access files were generated for the NumberSmasher using the Direct-to-Ascii and Ascii-To-Direct conversion software provided by DRES.

d. In the original UNIX version, program modules FORxASSIGN() are dynamically built, compiled, and linked at run-time using make files. Program complexity was reduced by hard-coding the contents of these modules and prompting the user for applicable data file names at run time.

Correct implementation of the required changes was tested using sample input and output files provided by DRES. To the limit of machine accuracy, NumberSmasher PC output for the test case provided was identical to the expected output.

### 4.3 Simplifying Assumptions

Several simplifying assumptions were incorporated in the revised FLISIM software, with the twin goals of reducing system complexity and increasing computational speed. The simplifications used are identical to those implemented in the predecessor to this contract:

a. Constant mass and inertial properties

FLISIM models the variation in HATT-X mass and inertial characteristics as rocket motors are consumed. For the neural network variant, these calculations were eliminated and replaced by the assumption of constant mass and inertia. The numerical values used were those present at the midpoint of the sample HATT-X profile provided by DRES:

mass = 217.4 kg

$x_{cm}$ = 1.858 m
$y_{cm}$ = $-4.2683 \times 10^{-10}$ m
$z_{cm}$ = $1.9116 \times 10^{-10}$ m

$I_{xx}$ = 20.26 kg $m^2$
$I_{yy}$ = 119.6 kg $m^2$
$I_{zz}$ = 136.8 kg $m^2$
$I_{xy}$ = $5.1609 \times 10^{-8}$ kg $m^2$
$I_{xz}$ = $2.0104 \times 10^{-8}$ kg $m^2$
$I_{yz}$ = $1.2884 \times 10^{-9}$ kg $m^2$

b. Constant thrust.

The boost-glide flight profile of the HATT-X vehicle was replaced by a constant thrust model. During the first two seconds of flight, the vehicle thrust is 25,000 N. This values ensures that sufficient airspeed is present upon leaving the rail. At all subsequent times, the thrust is constant at 1,358 N. This value was used for work conducted under the predecessor contract, and is the trim thrust required for level flight at 10, 000 metres altitude and 0.6 Mach.

c. Two-dimensional flight

Task II examined the application of neural network methods to pitch control of the HATT-X vehicle. Further simplification was implemented by limiting the vehicle to 2D flight. This condition was imposed by setting the aileron incidence to be identically zero, disabling the wind elements of the atmospheric model, and assuming a zero thrust moment vector.

## 5.0 Longitudinal Neural Network Development

### 5.1 Training Data Generation

Generation of neural network training data was motivated by analogy to the learning procedure of a hypothetical human operator. The human learning model assumed is described below, and is followed by a description of the corresponding computer implementation.

The model assumes a student pilot placed in the cockpit with no prior knowledge of aircraft dynamics. The pilot knows the current state of the aircraft (from instruments and out-the-window views) and the desired aircraft state, but possesses no understanding of the procedure used to achieve the terminal state. As a result, his only option is to manipulate the controls and observe the aircraft's response. With time, the student learns the association between control inputs and the aircraft's response.

While learning the "feel" of the aircraft, the student executes commands which are essentially random. During this process, he may be expected to execute actions which endanger the survivability of the vehicle. To avoid this situation, an instructor pilot is provided. The instructor maintains a "laissez-faire" attitude, remaining quiet until the aircraft is in risk, at which time he brings the aircraft back under control before stepping into the background.

This model may be implemented by the following procedure:

1. At time $t$, the vehicle is in state $x_0 = x(t)$. The pilot notes $x_0$ and randomly selects a control input $\delta$.

2. The instructor evaluates the vehicle state $x_0$. If it is found to be unsafe, the student command $\delta$ is overridden by an instructor-generated command $\delta^*$.

3. The control input ($\delta$ or $\delta^*$) is maintained for a period of $\Delta t$ seconds. The final state $x_f = x(t+\Delta t)$ is noted.

4. The pilot learns that the transition from $x_0$ to $x_f$ requires a control input of $\delta$ (or $\delta^*$).

5. Repeat from step 1.

Generation of training data was conducted in accordance with this procedure. Data collection commenced two seconds after launch. The initial state was comprised of the vector $[\ h, M, \alpha, q, \theta\ ]$. The final state was a single element vector consisting of the final flight path angle $\gamma$. Commanded canard deflection $\delta$ was limited to the range

$-5° \leq \delta \leq 10°$. The period $\Delta t$ between measurement of initial and final states was set to one second.

The "instructor" was implemented by modifying the randomly generated value of $\delta$ according to the following rules:

1. $\delta = 15°$ if $h < 300$ m or $\theta < -30°$.
2. $\delta = -15°$ if $h > 4000$ m or $\theta > 30°$.

Training data records were not written for cases in which the instructor logic was activated. This decision was made to ensure that the distribution of cases in the training data set was not skewed toward extreme limits of the input values.

A final modification to the data generation model was implemented to ensure that the generated data did not cluster in one region of the input space. On each sample, the canard deflection $\delta$ had a 1 in 20 probability of being set to $\pm 15°$. This command value causes a rapid pitch up or down of the aircraft to a new region of the input space. As in the case of instructor-generated commands, a training data record was not written when this logic was activate.

Using this algorithm, ten thousand training examples were collected over the course of several flights. The data was checked and found to be well distributed over the input space of the neural network.

This data set was used to train several backpropagation networks of varying size. Poor network convergence was observed in all cases, despite increasingly large network sizes which approached 100 hidden neurons. After examination of these results, Atlantis hypothesized that the network was not being presented with sufficient input data. Reexamination of the vehicle dynamics indicated that the canard servo lag time modelled was approximately 0.5 seconds. This lag is of the same order of magnitude as the 1.0 second delay between control application and state measurement used in generation of training data, and is therefore significant. Due to this consideration, a second set of network training data was generated. The network input vector was augmented by including $\delta_0$, the actual canard deflection present before application of the new canard command. Network convergence during offline training was greatly improved by this modification.

### 5.2 Network Training Results

Several backpropagation networks were trained using the data generated in the previous section. The generated input-output examples were divided into two sets for training and test purposes. The training data set consisted of 2332 examples, while the test data set incorporated 3331 records.

The network architectures tested consisted of several three and four layer networks. All networks had seven inputs and one output, with the input/output vector formatted according to Table 5.1.

**Table 5.1:** IDM Network Input and Output Formats

| Input/ Output | Index | Variable | Scaling |
|---|---|---|---|
| Input | 1 | Altitude | -4000 to 4000 m |
| Input | 2 | Mach number | -1.0 to 1.0 Mach |
| Input | 3 | Angle of attack | -10.0° to 10.0° |
| Input | 4 | Pitch rate | -30°/s to 30°/s |
| Input | 5 | Pitch angle | -40° to 40° |
| Input | 6 | Current canard deflection | -12° to 12° |
| Input | 7 | Desired flight path angle | -50° to 50° |
| Output | 1 | Commanded canard deflection | -12° to 12° |

Training results for several four layer networks are presented in Table 5.2. The four layer networks provided superior performance over three layer networks with an equivalent number of weights. Results for the three layer networks have not been included in this report.

On the basis of these results, a 7-10-5-1 architecture was selected to provide the basis for future work. This network was judged to provide the best tradeoff between output accuracy and network size. Off-line training convergence results for this network are illustrated in Figure 5.1.

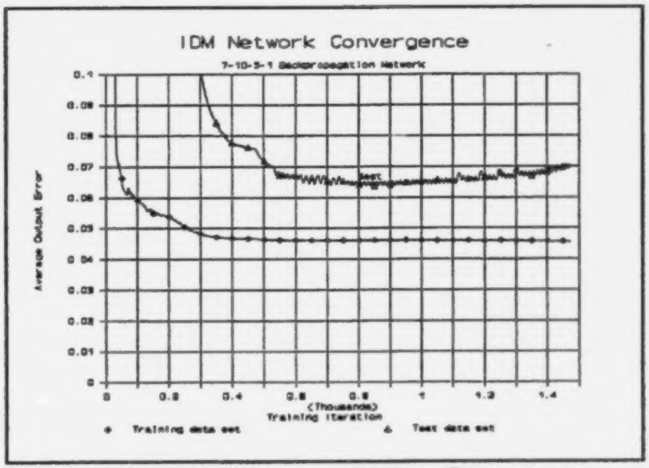

**Figure 5.1:** IDM Network Offline Convergence

**Table 5.2:** IDM Network Convergence Results

| Network Architecture | Average test set error | Maximum test set error | # of connections |
|---|---|---|---|
| 7-20-20-1 | 0.058 642 | 0.322 311 | 601 |
| 7-15-15-1 | 0.061 549 | 0.269 919 | 376 |
| 7-10-10-1 | 0.059 021 | 0.282 852 | 201 |
| 7-5-5-1 | 0.072 752 | 0.303 536 | 76 |
| 7-10-5-1 | 0.060 012 | 0.269 225 | 141 |
| 7-5-10-1 | 0.080 030 | 0.315 449 | 111 |

### 5.3 Network Performance

To evaluate its performance, the IDM neural network generated was integrated with the MPS-100 autopilot model in the FLISIM software. The standard MPS-100 autopilot implements an altitude hold control law whose format is illustrated in Figure 5.2. The IDM network was substituted in place of the PID inner loop controlling the vehicle's flight path angle. The resulting architecture is illustrated in Figure 5.3. By performing this "black box" substitution, comparison of the neural network's performance with the traditional control strategy was easily accomplished.

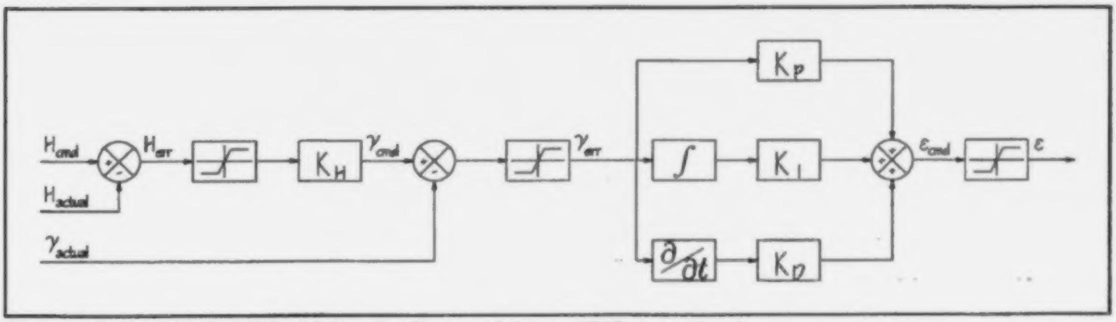

**Figure 5.2:** MPS-100 Altitude Control Law (PID version)

Several sample flights were conducted to determine the performance characteristics of the neural network controller. The static accuracy of the network was evaluated by disabling online learning for these trials.

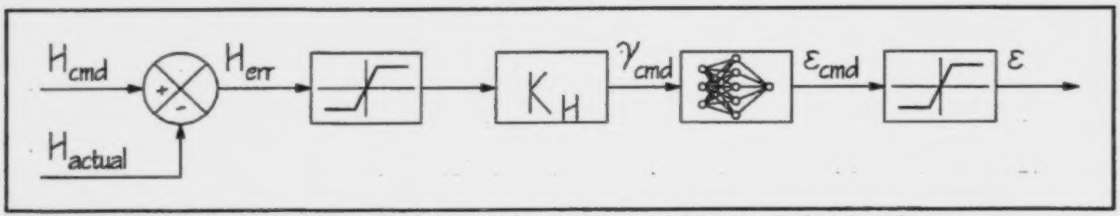

**Figure 5.3:** MPS-100 Altitude Control Law (Neural Network Variant)

Each sample flight consisted of a series of altitude hold commands, with step transitions between the commanded values. To prevent excessive loss of airspeed, each step was constrained to be 1000 m or less. Each flight was conducted once with the traditional PID controller in place and once with the inner control loop replaced by the IDM network. Height profiles and canard command values are depicted in Figure 5.4 through Figure 5.7 for two sample flights. The sequence of commanded altitudes for each flight is tabulated in Table 5.3.

**Table 5.3:** Commanded Altitude Profiles for Neural Network Evaluation Flights

| Flight Number | Commanded Altitudes (metres) | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1000 | 1500 | 1000 | 2000 | 1500 | 2000 | 3000 |
| 2 | 1000 | 2000 | 1500 | 1000 | 2500 | 1500 | 500 |

Two generalizations may be made on the basis of the altitude and control deflection versus time graphs. It is quickly apparent from the altitude profiles that the neural network controller produces performance substantially similar to that obtained with the PID inner loop. A more interesting observation arises upon examination of the control deflection histories. The neural network is seen to achieve its goal by applying a relatively smaller control deflection for a slightly extended period of time. As a result, the neural network controller effects a smoother transition between pitch attitudes at the expense of a somewhat increased lag time.

These results indicate that the neural network controller is able to track commanded pitch attitudes generated by the longitudinal control laws. Under normal flight conditions, the controller performance is equivalent but not superior to that obtained using the existing PID control laws. Substantial performance improvement should not be expected however, as the vehicle trajectory commanded by the controller outer loop is identical for both the PID and NN implementations of the controller inner loop.

Substantial increases in performance will require a redesign of the controller outer loop. This may be accomplished in various ways. First, an additional neural network may be used to generate the flight path angle commands. Second, the existing controller structure may be optimized using genetic algorithms (as evidenced by the experimental results of Section 3.0 of this document, and Section 5.0 of the contract interim report[3]).

These arguments would seem to indicate that no benefit is derived by replacing the PID inner loop of the flight controller with a neural network. This conclusion is not true. By adaptively updating the neural network during flight, it is possible to develop a vehicle controller which automatically responds to and corrects for off-nominal vehicle characteristics or malfunctions. This ability was demonstrated in Section 4.0 of the interim report[3]. Results obtained using online training of the FLISIM neural network controller are documented in the following section.

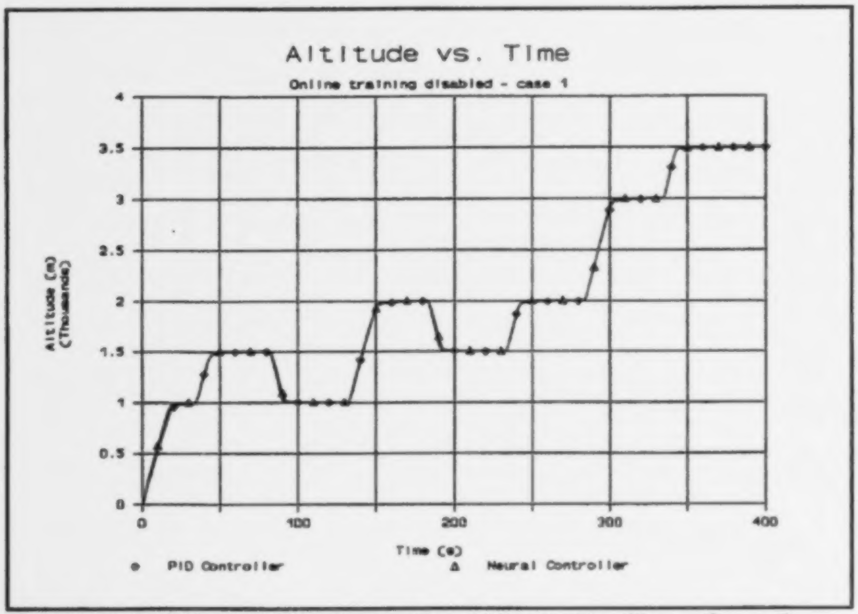

**Figure 5.4:** PID and NN Controller Altitude Profiles - Case 1

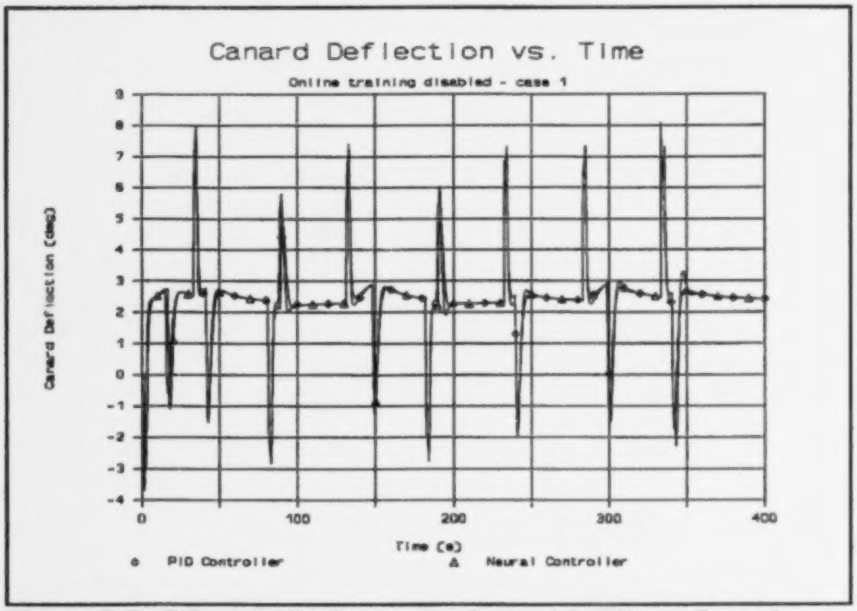

**Figure 5.5:** PID and NN Controller Canard Profiles - Case 1

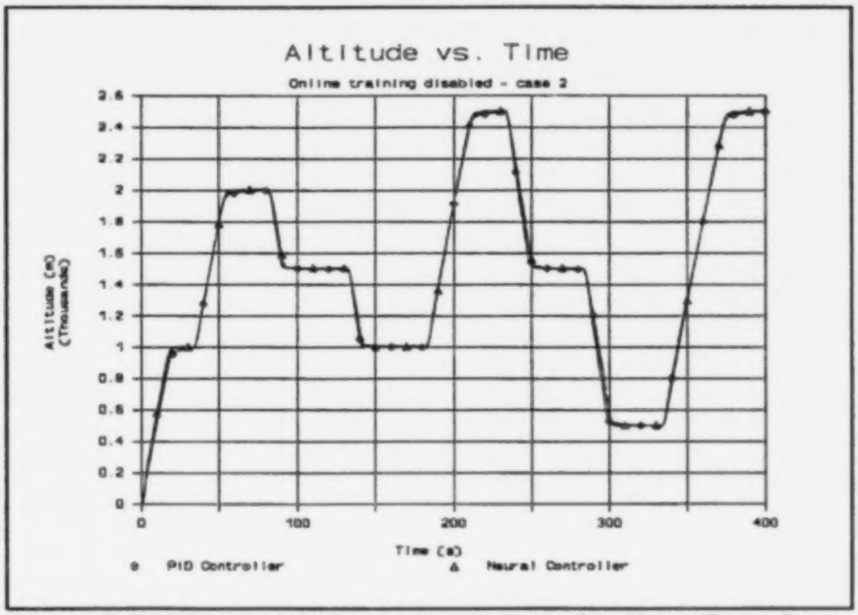

**Figure 5.6:** PID and NN Controller Altitude Profiles - Case 2

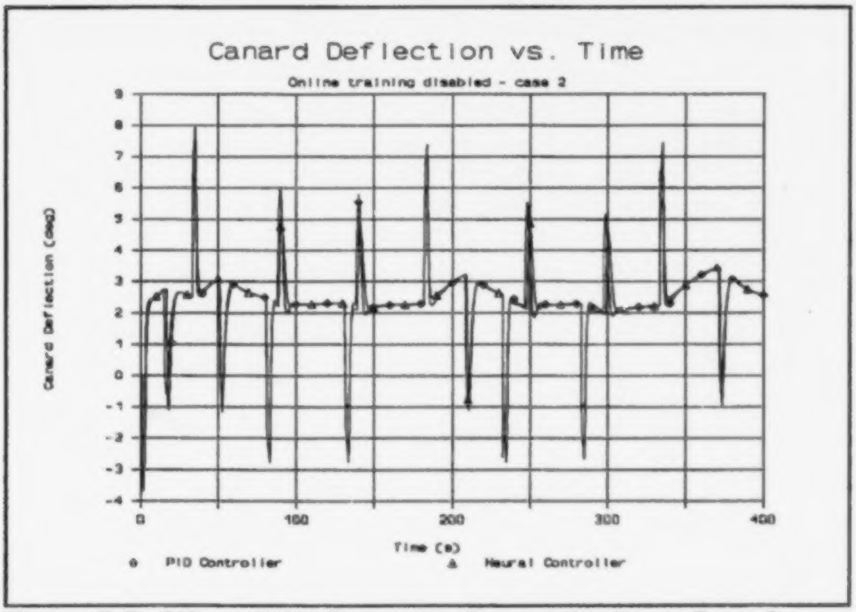

**Figure 5.7:** PID and NN Controller Canard Profiles - Case 2

### 5.4 Online Training Results

Several test flights were conducted with online training of the IDM network enabled. In contrast with the results observed in the Phase I work, online training was seen to decrease controller performance. A sample flight profile with online training enabled is provided in Figure 5.8. In the preceding section, the PID and neural network (without online training) controllers were shown to produce reasonably consistent results throughout the flight profile. When online training is enabled, the neural controller gradually loses its ability to track the commanded value, leading to eventual loss of control.

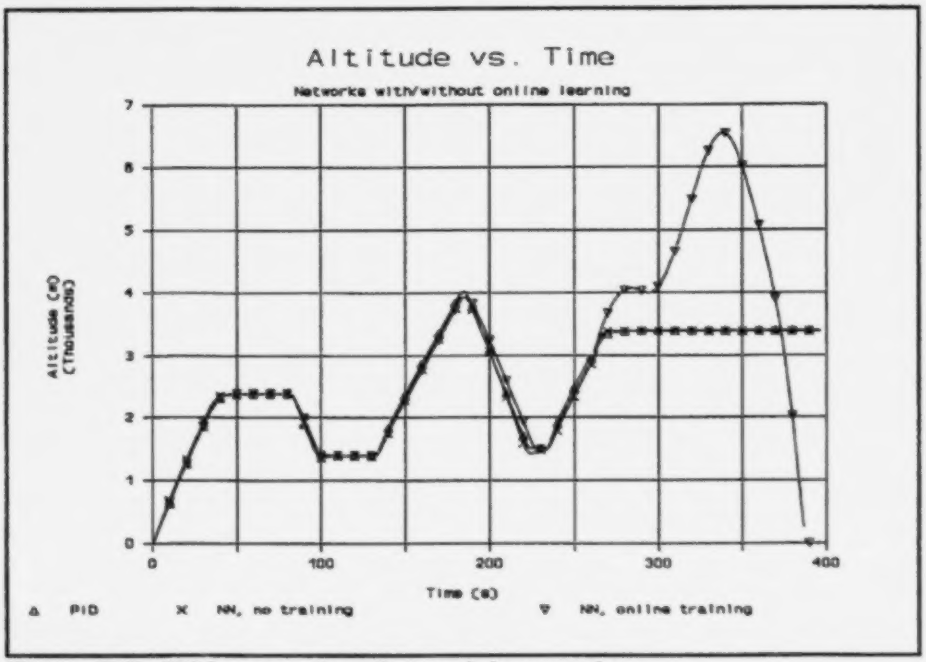

**Figure 5.8:** IDM network online training results

Atlantis believes that the gradual failure of the IDM network controller is due to repeated training in a small section of the network input space. Inspection of Figure 5.8 shows that throughout the majority of the flight, the vehicle is in a condition of straight and level flight. During these segments, online learning increases the network's ability to perform straight and level flight, at the expense of reduced accuracy in other areas of the input space. As training proceeds, the output in these other regions diverges significantly from its correct value. When a waypoint transition occurs, the network input leaves the region of straight and level flight and enters a region of potentially erroneous output. Correspondingly, the vehicle abandons its commanded trajectory. Disaster may occur unless the network input subsequently enters a region of increased accuracy.

The decrease in overall network performance due to overtraining in one area of the input space was noted by Atlantis in the predecessor to this contract[4]. At that time, it was thought that no serious implications would result, as the time history of the network input vector would evolve gradually enough to maintain a locally-centred region of high output accuracy. In cases where the input suddenly jumped to a new region, online learning would improve the accuracy after a few cycles so that vehicle controllability was not seriously impaired. This hypothesis was supported by results discussed in the interim report[3].

Further analysis by Atlantis has shown that this hypothesis is not generally valid as convergence of online training to a locally optimum solution is not guaranteed. Online learning takes place in an input region which is not necessarily coincident with that corresponding to the current state. The proof for this statement is straightforward and is presented below:

1. Assume that at time $t$ the vehicle is in initial state $x_t$, with commanded flight path angle $\gamma_t$.
2. Corresponding to the input vector $(x_t, \gamma_t)$, the network produces a canard deflection $\delta_t$.
3. The network-generated command $\delta_t$ results in a vehicle state at time $t+\Delta t$ of $x_{t+\Delta t}$, with actual flight path angle $\gamma_{t+\Delta t}$.
4. The network weights are updated so that the input vector $(x_t, \gamma_{t+\Delta t})$ produces output $\delta_t$. This training is assumed to improve output accuracy over a region $\beta_1$ centred at the training input vector $(x_t, \gamma_{t+\Delta t})$.
5. The network input vectors $(x_t, \gamma_t)$ and $(x_t, \gamma_{t+\Delta t})$ are separated by a distance $\beta_2 = (\gamma_{t+\Delta t} - \gamma_t)$.
6. If $\beta_2 > \beta_1$, output accuracy at the flight input vector $(x_t, \gamma_t)$ is not improved.

As an extreme example of this problem, consider a vehicle which has been in straight and level flight for an extended period of time. During this period, the network has been repeatedly trained with mapping examples in a small region centred around the trim state. Further assume that as a result of this training the network output values closely approximate the current trim setting, no matter what the input state.

The vehicle is now issued a command to pitch up. However, it outputs the canard setting for level flight, as it has repeatedly been taught to do. As a result, the vehicle does not change its attitude. Following the training procedure specified for this controller structure, the network is trained that commanding the trim canard setting when in level flight will result in level flight. Essentially, the network does not learn anything new, as this is the same pattern it has been repeatedly seeing. The cycle now repeats. Clearly, the vehicle will continue in level flight indefinitely, no matter how large a pitch command is issued.

On the basis of these arguments, Atlantis was not able to guarantee that online learning would improve controller performance a given vehicle state. Attention was therefore focused on diminishing the weight-distorting effects of online learning. Three potential solutions to this problem were considered. Each of these is documented in the following sections.

### 5.4.1 IDM Error Network Architecture

This approach to online learning was based on two tenets:

a. Since global weight distortion occurs due to repeated training in one local area, training should be halted once an acceptable error level is achieved. This imposes an upper limit on the number of training cycles at a given input state.

b. Before online training occurs, the neural network approximates the vehicle IDM across all attainable states. Significant errors do not arise until online training is active. The controller structure can therefore be altered to use two neural networks, as illustrated in Figure 5.9. The IDM network is trained offline using the normal procedure. Its weights are then frozen, and are not updated during flight. During flight, a second network is trained on the error between the IDM mapping and the actual vehicle response. At each time step the canard deflection commanded is the sum of the IDM and error network outputs.

   The error network output is limited to a much smaller output range than that of the IDM network. As a result, it may be used to "tune" the output of the IDM network, but can not completely override its output. This structure is expected to improve local performance while limiting the magnitude of any global errors which develop as a result of online learning.

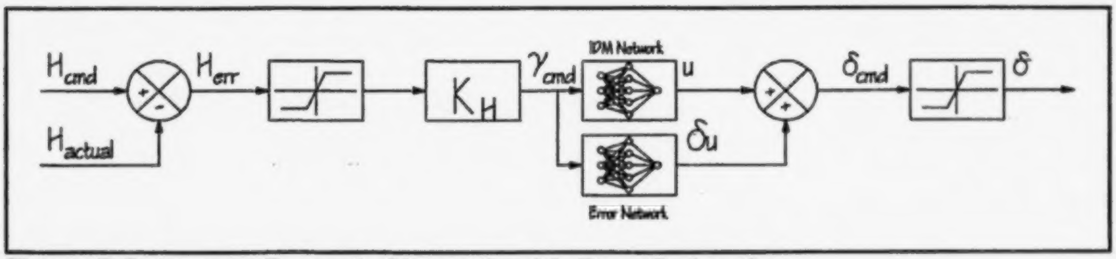

**Figure 5.9:** Inverse Dynamic Network with Error Estimation

Each of these modifications were implemented on the neural controller. The error network output was scaled to the maximum error encountered during training of the

IDM network. By reference to Table 5.2, the maximum output of the error network is seen to be 22.53 % of the corresponding IDM network value.

Figure 5.10 demonstrates the performance improvement achieved through applying these modifications to the online learning algorithm. Flight trajectories generated using the PID controller and IDM network alone are included for comparison. With the addition of the error network, the vehicle remains controllable throughout the flight profile. However, performance is significantly inferior to that obtained from the PID controller.

Due to its relatively poor performance, the error network architecture was not pursued beyond these trials.

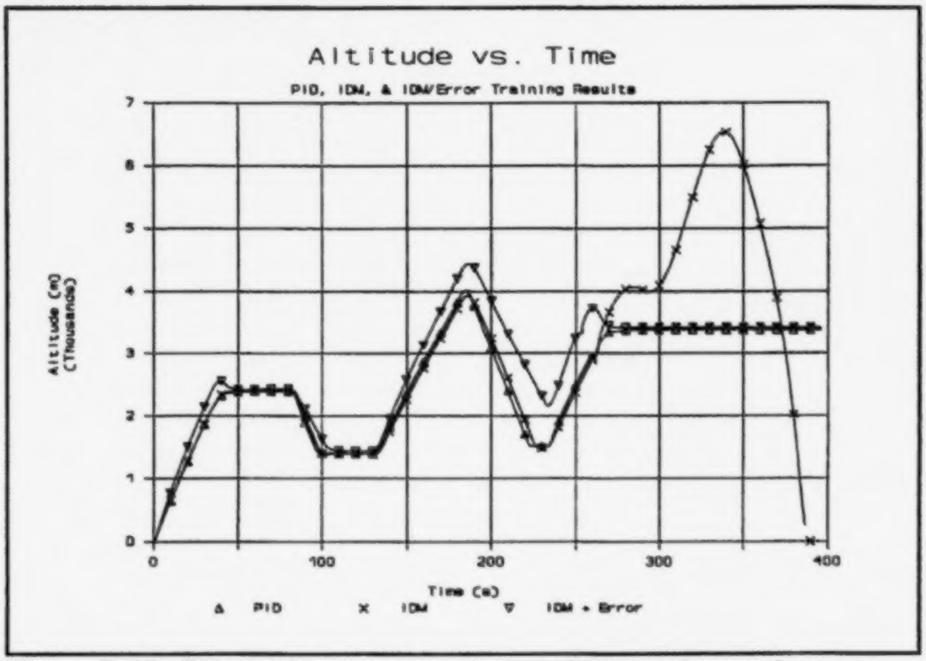

Figure 5.10: Effect of error network on online training performance

### 5.4.2 Inverse/Forward Dynamic Model

The architecture of this controller is illustrated in Figure 5.11. As in the IDM controller model, the error between commanded ($H_{cmd}$) and actual ($H_{actual}$) altitude is propagated through the outer loop and the IDM network to produce (respectively) a commanded flight path angle $\gamma_{cmd}$ and canard deflection $u$. Unlike the IDM controller, however, $u$ is not fed directly into the servos. Instead, the current vehicle state and the commanded canard value $u$ are used as inputs to a second network. This network

**models the forward dynamics of the vehicle, producing an estimate $\gamma_{predicted}$ of the actual flight path angle which would result from application of $u$.**

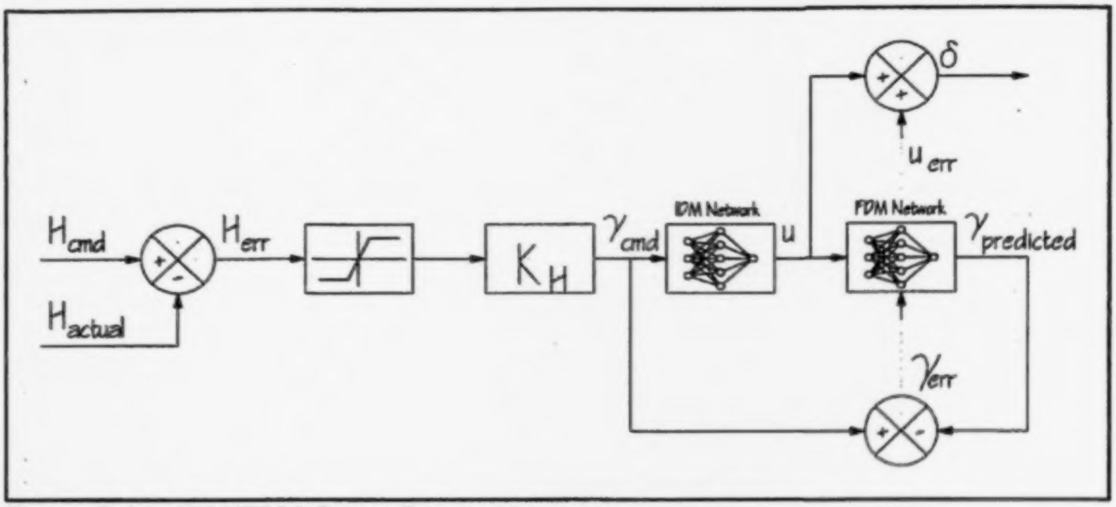

**Figure 5.11:** IDM/FDM Controller Architecture

In general, the values of $\gamma_{cmd}$ and $\gamma_{predicted}$ will not be identical. The difference between these values is denoted by $\gamma_{err}$. By interpreting this value as an error signal, it is possible to backpropagate it through the forward dynamic model network, producing a corresponding error estimate $u_{err}$ at the input layer of the network. Other researchers have shown that $u_{err}$ serves as an estimate of the amount by which $u$ must be increased in order for the FDM network output to increase by $\gamma_{err}$.

Under the assumption that the FDM mapping is correct, the commanded canard setting $u$ is therefore increased by the quantity $u_{err}$. This value ($\delta$) forms the actual control setting output to the servos.

During online training, the IDM network is updated in the usual manner. The FDM network mapping is also updated by mapping the initial state and canard deflection $\delta$ to the resulting vehicle state.

Experimental results obtained with this architecture are illustrated in Figure 5.12. Performance is seen to be comparable to that obtained with the PID controller. In addition, no stability problems were evidenced in the course of several test flights.

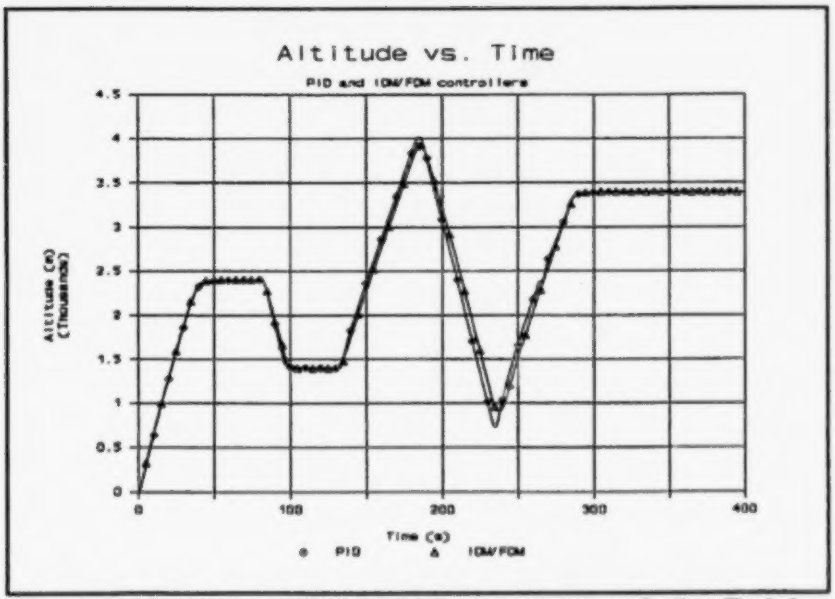

**Figure 5.12:** Effect of IDM/FDM Architecture on Online Training Performance

Unfortunately, the IDM/FDM architecture possesses a very high computational complexity relative to the standard IDM architecture. Using an IDM/FDM structure, four forward propagations, three back propagations, and two weight updates are required per autopilot control cycle. This is a significant increase over the IDM architecture, which requires two forward propagations, one back propagation, and one weight update. The ability of the controller to execute in real time on an actual flight vehicle may be compromised by this additional computation. This subject is further addressed in Section 11.

### 5.4.3 IDM Network with Proportional Training

Unlike the theoretical elegance of the preceding architecture, this controller structure presents a simplified format relative to the IDM architecture. The physical arrangement of control elements in this structure is identical to the baseline IDM architecture. Instead, the method used to conduct online training is varied.

As with the standard IDM architecture, the proportional training variant uses the IDM network to produce a desired canard setting $\delta$ on the basis of the current vehicle state $x_0$ and the desired flight path angle $\gamma_{cmd}$. One second later, the resulting flight path angle $\gamma_{actual}$ is measured.

At this point, the two architectures diverge. Using the baseline method, the training example presented is the mapping $(x_O, \gamma_{actual}) \Rightarrow (\delta)$. In contrast, the proportional training method trains on the mapping

$$(x_0, \gamma_{cmd}) \Rightarrow (\delta + \epsilon) \tag{5.1}$$

where

$$\epsilon = k_1 (\gamma_{cmd} - \gamma_{actual}) \tag{5.2}$$

and $k_1$ is an appropriately determined constant of proportionality. In this research, $k_1$ was arbitrarily set to 0.25, which produced adequate learning performance. Ideally, however, $k_1$ should be based on an analysis of the individual vehicle dynamics:

$$\left[\frac{\partial \dot{\gamma}}{\partial \delta}\right] (\partial \delta) \Delta t = \epsilon \tag{5.3}$$

$$\partial \delta = \frac{\epsilon}{\left[\frac{\partial \dot{\gamma}}{\partial \delta}\right] \Delta t} \tag{5.4}$$

$$k_1 \epsilon = \frac{\epsilon}{\left[\frac{\partial \dot{\gamma}}{\partial \delta}\right] \Delta t} \tag{5.5}$$

$$k_1 = \left[\frac{\partial \dot{\gamma}}{\partial \delta} \Delta t\right]^{-1} \tag{5.6}$$

Several points should be noted about this architecture:

1. Unlike the baseline IDM method, this controller is always trained at the current state $(x_O, \gamma_{cmd})$. As a result, controller stability is enhanced.

2. This architecture implicitly incorporates prior knowledge about the process being modeled (ie: that a flight path angle less than its commanded value requires a more positive canard deflection). Online performance is improved by incorporating human knowledge in the training process. In contrast, the

FDM/IDM architecture relies on a limited accuracy model of the dynamic system (the FDM) to determine appropriate adjustments to the control settings.

Sample flight profiles for the Proportional Training IDM are presented in Figure 5.13 and Figure 5.14. These figures plot altitude and flight path angle error histories for the IDM architecture with and without online proportional training. The ability of the Proportional Training algorithm to maintain stable and accurate command tracking is evident from these plots.

As with the other methods of online training, the Proportional Training algorithm became unstable and saturated the output neurons when large learn rates were used. However, the controller networks were found to be stable over a much larger range of learn rates. Using the Proportional Learning algorithm, learn rates up to 0.05 were tested and found to produce stable outputs. In contrast, the baseline IDM algorithm saturated the network outputs for learn rates above 0.02.

On the basis of these results, the IDM Network with Proportional Training architecture was selected for further evaluation.

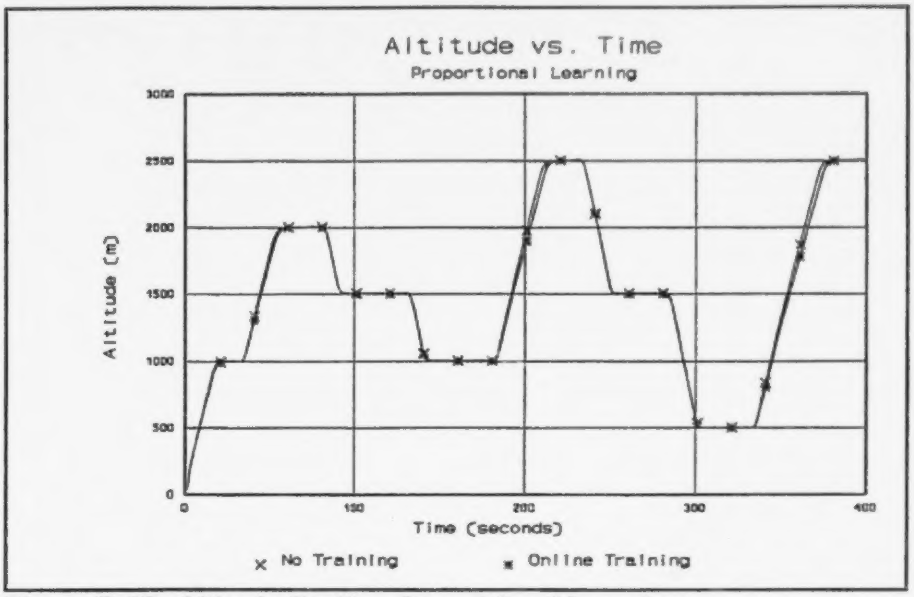

Figure 5.13: Altitude Profile - Proportional Learning

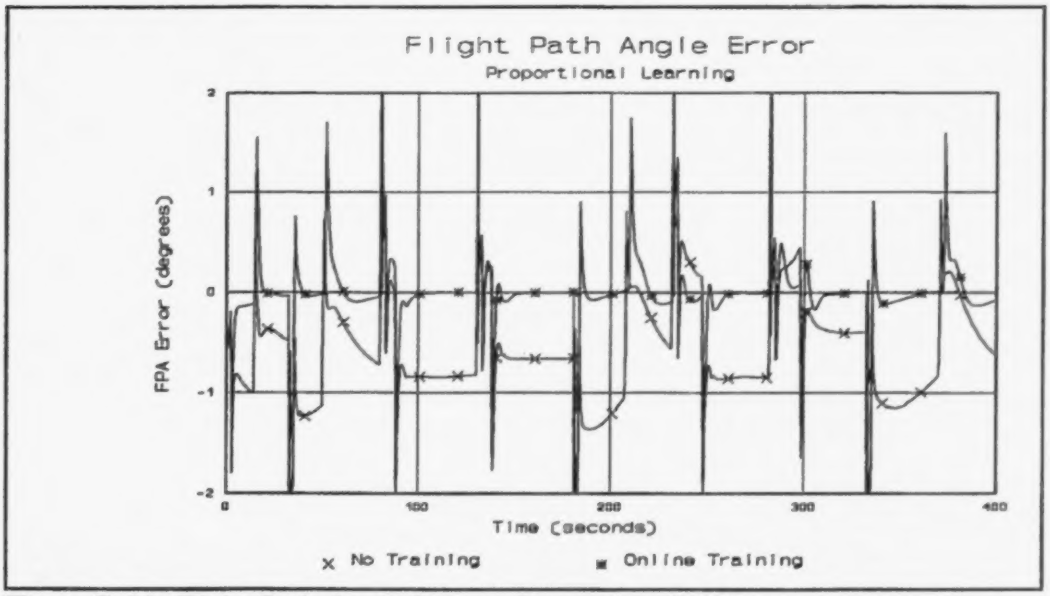

Figure 5.14: Flight Path Angle Errors - Proportional Training

## 6.0 Application to Off-Nominal Vehicle Configurations

The adaptive ability of the neural network controller developed in Section 5 was tested through control of a number of off-nominal vehicle configurations. The test configurations selected are similar to those used for off-nominal evaluation of the simple flight model controller in Section 4.0 of the interim report[3]. These configurations are summarized in Table 6.1.

**Table 6.1:** Off-Nominal Vehicle Configurations

| Configuration | Description |
|---|---|
| Servo offset | Actual canard offset is two degrees greater than the value commanded by the neural controller. |
| Reduced servo effectiveness | Actual canard offset is half the value commanded by the neural controller. |
| Increased inertia | The vehicle mass and inertia are increased by 25 % over their nominal values. |

## 6.1 Off-Nominal Case 1 - Servo Positioning Offset

For this test case, the deflection command issued to the canard servo is two degrees greater than the actual value commanded by the HATT-X autopilot. This case simulates the controller's ability to adjust to potential vehicle integration errors, such as component misalignment, incorrect calibration, or improper servo nulling.

Figure 6.1 depicts the altitude profiles which result for this test case. Altitude histories for the traditional PID controller, IDM network, and IDM network with proportional training are included in this figure. Figure 6.2 illustrates the improved tracking ability of the IDM network when online learning is enabled.

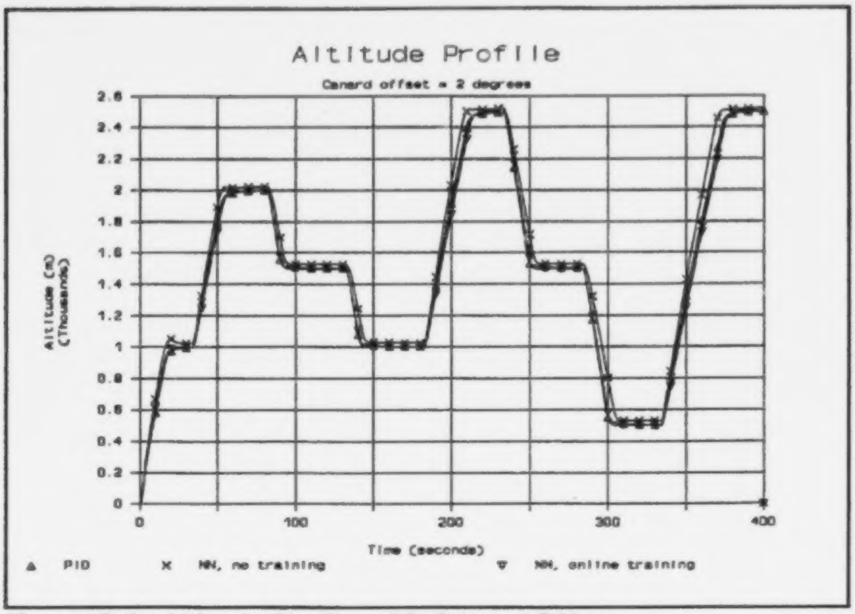

**Figure 6.1:** Altitude Profile - 2° Canard Offset

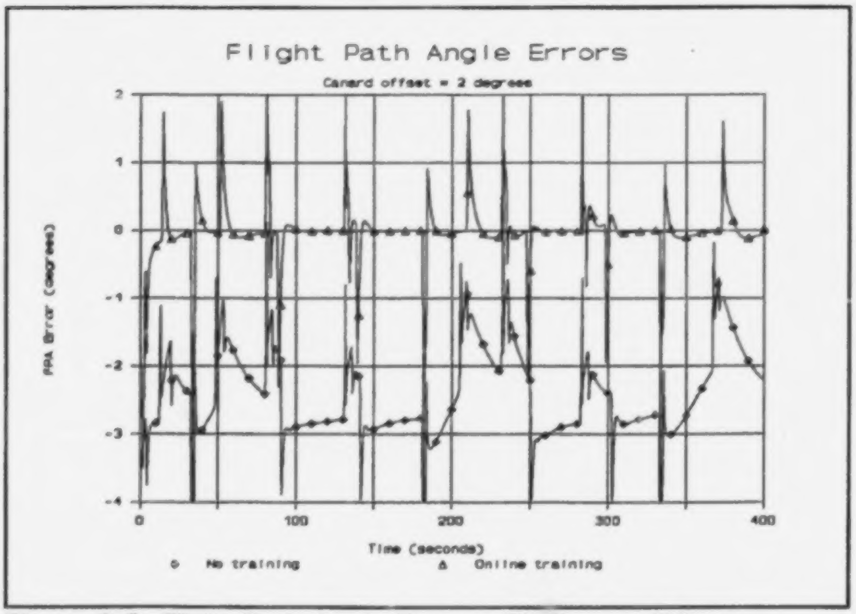

**Figure 6.2:** Flight Path Angle Errors - 2° Canard Offset

## 6.2 Off-Nominal Case 2 - Reduced Canard Effectiveness

For this test case the canard deflection output to the servos was set to 50 % of the value commanded by the controller. Again, the neural autopilot is able to compensate for the diminished controllability of the vehicle.

Flight profiles for this case may be found in Figure 6.3 and Figure 6.4. Significant oscillatory and overshoot tendencies are apparent in the untrained neural controller. These characteristics are somewhat reduced for the PID controller, but are virtually eliminated once online training of the neural controller is activated.

## 6.3 Off-Nominal Case 3 - Increased Mass and Inertia

For this test case the airframe's mass and inertia characteristics were increased to 125% of their nominal values. Again, the neural network controller with online training provides the best performance, as evidenced by Figure 6.6.

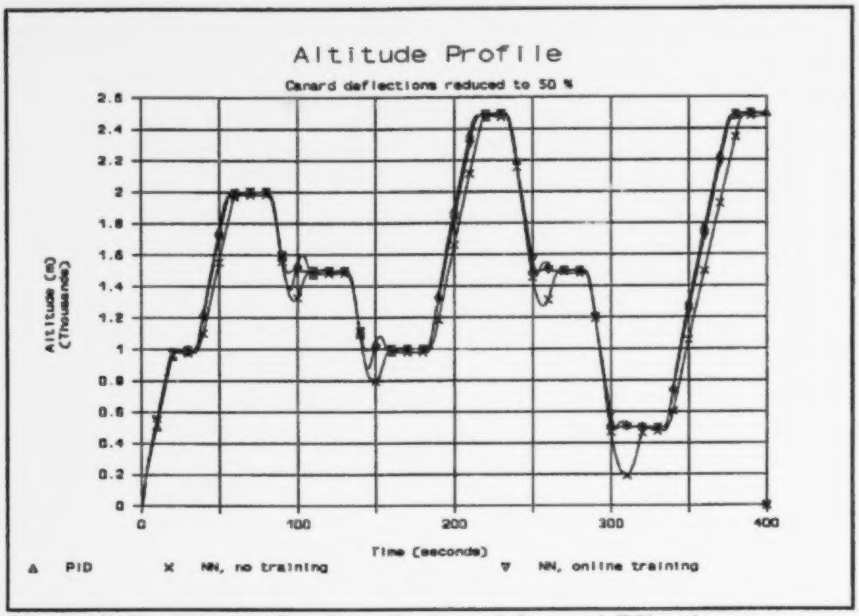

**Figure 6.3:** Altitude Profile - Reduced Canard Effectiveness

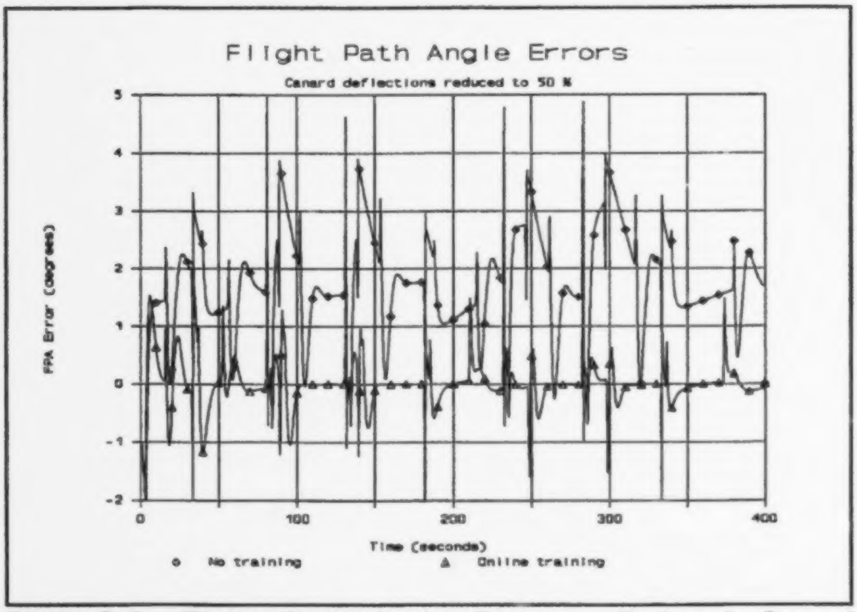

**Figure 6.4:** Flight Path Angle Errors - Reduced Canard Effectiveness

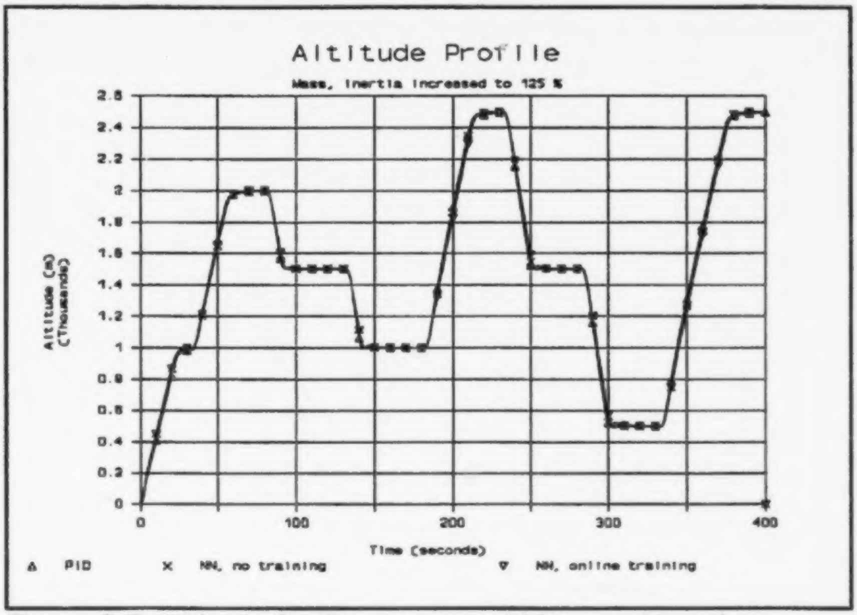

**Figure 6.5: Altitude Profile - Increased Mass and Inertia**

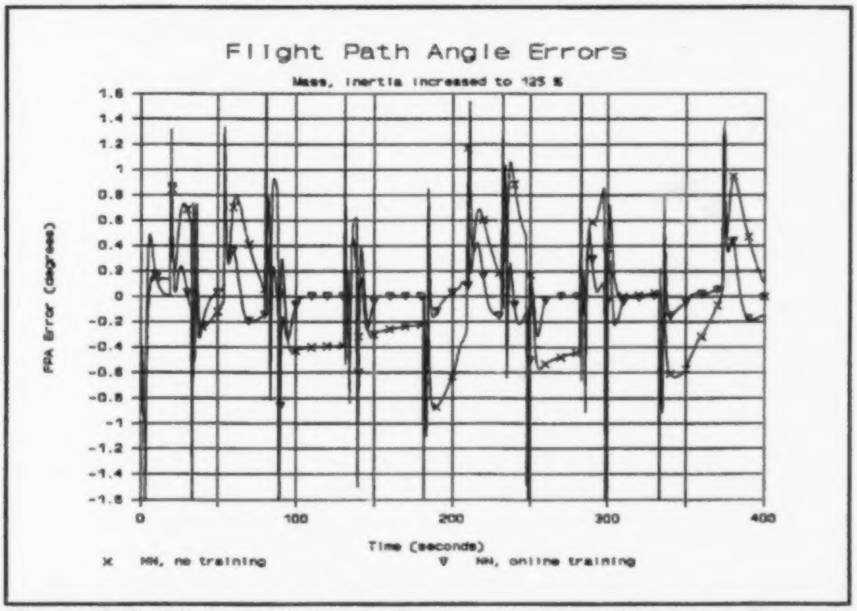

**Figure 6.6: Flight Path Angle Errors - Increased Mass and Inertia**

## 7.0 Application of Genetic Algorithms to Longitudinal Controller Optimization

### 7.1 Introduction

Sections 5.0 and 6.0 of this report demonstrated the ability of neural networks to introduce an adaptive component in a traditional control law. Comparison with the original PID control laws shows that the neural network controller provides equivalent performance under nominal operating conditions and improved tracking under degraded mode operations. These performance improvements were achieved without the need to incorporate detailed knowledge of the vehicle dynamics.

Having verified the suitability of the neural network for use as an inner loop control element, Atlantis shifted its attention to design of the outer control loop.

Based on the results of Section 5.3 of the interim report[3], and Section 3.0 of this document, Atlantis decided to apply genetic algorithms to optimization of the PID controller outer loop gains. Use of a second neural network in place of the existing outer loop structure could also have been attempted. However, this was rejected due to substantial difficulties encountered in previous applications of this technique[3,5].

### 7.2 Genetic Algorithm Implementation

Operation of the genetic algorithm was discussed in Section 5.3 of the interim report[3], and will not be repeated here. Instead, the parameters selected for optimization and the evaluation criteria used will be discussed.

As stated earlier, no modifications were made to the structure of the controller outer loop. Only the numerical values used for gain and limit settings were adjusted. The selected parameters and their usage are detailed in Table 7.1. Further discussion of each term and its significance may be found in the ROBOT-X software specifications[6].

Each generation of the genetic algorithm consisted of 200 individuals. Initial gene values for each individual were randomly chosen. Gene values for all individuals were constrained to lie between zero and ten times their nominal values.

Each individual generated was tested on a sample mission profile consisting of several altitude transitions over a period of 180 seconds, as shown in Table 7.2. The performance score assigned to each individual was the time integral of the actual altitude error, as given in (7.1).

$$score = \int_0^{180} \left| h_{commanded} - h_{actual} \right| dt \qquad (7.1)$$

**Table 7.1:** Longitudinal Control Law Parameters Altered by Genetic Algorithm

| Parameter | Nominal Value | Description |
|---|---|---|
| K3A | 0.2164 | Mach-proportional component of longitudinal stability augmentation gain |
| K3B | -0.226 | Mach-independent component of longitudinal stability augmentation gain |
| KH | $1.7 \times 10^{-3}$ | Altitude error to flight path angle error proportional gain |
| KA11A | 4.7467 | Mach-proportional component of flight path error-proportional canard gain |
| KA11B | 0.3745 | Mach-independent component of flight path error-proportional canard gain. |
| $KA11_{MAX}$ | 2.600 | Maximum permissible value of KA11 |
| G11 | 0.150 | Flight path error-integral canard gain |
| GACOLI | 5.000 | Flight path angle error limiting value |

**Table 7.2:** Altitude Profile for Evaluation of Genetically Derived Controllers

| Time Period (seconds) | 0-30 | 30-80 | 80-130 | 130-180 |
|---|---|---|---|---|
| Commanded Altitude (metres) | 500 | 1000 | 2000 | 500 |

### 7.3 Genetic Algorithm Results

Using the experimental arrangement of Section 7.2, the genetic algorithm was executed and allowed to proceed for 28 generations before termination. Testing of the 5600 controllers generated required 17 days of continuous execution on the i860 processor.

The average and best scores for each generation are illustrated in Figure 7.1 and Figure 7.2. The evaluation score obtained by the baseline PID controller is also included in these figures to allow direct comparison.

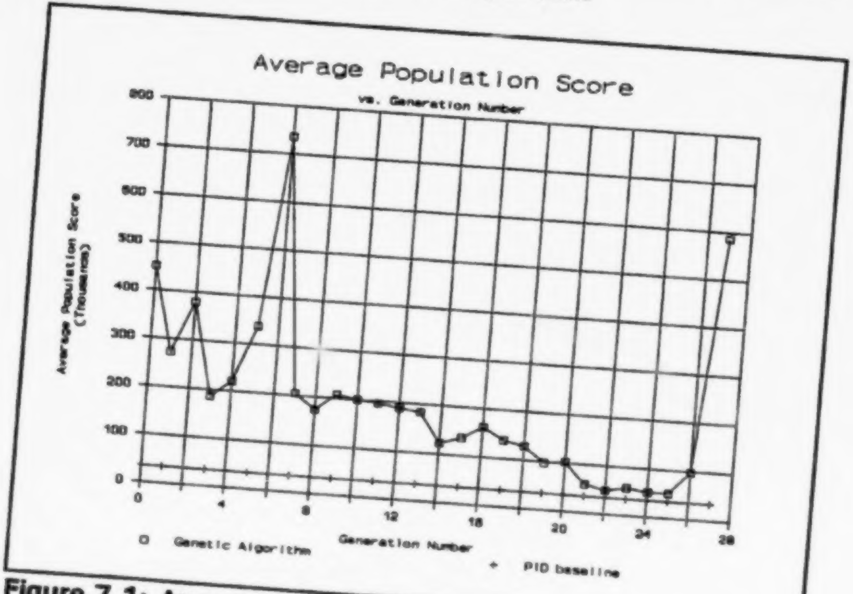

Figure 7.1: Average Population Scores

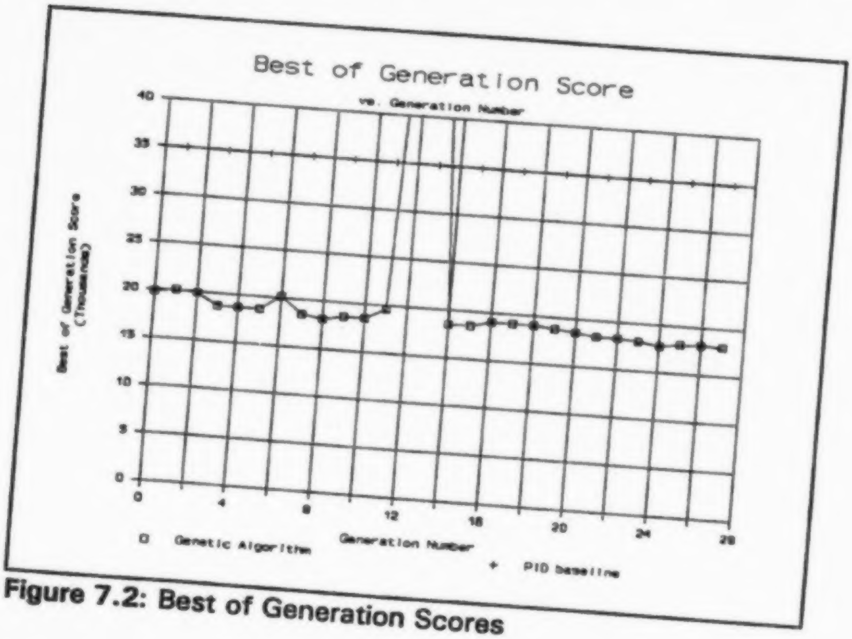

Figure 7.2: Best of Generation Scores

Figure 7.1 shows the average population score to be generally decreasing, although large perturbations in the curve are apparent. The general improvement in performance is to be expected from the selective reproduction used in the genetic algorithm. Spikes in the curve reflect the occasional situation in which the effects of mutation have a negative effect on the general suitability of the population.

The average individual score is also seen to approach but not surpass that received by the baseline PID controller. This relatively poor performance is to be expected, as each generation includes several individuals with exceedingly poor characteristics. The performance of the population as a whole is not of great importance, however, as we are primarily interested in the performance of the best of generation individual. The sole use of the average score for each generation is as an indicator of process completion. The genetic algorithm should generally be run until no further improvement in the average population occurs. Such a situation indicates that the algorithm is very close to finding the optimal solution, and further efforts are not likely to yield improved results. On this basis, GA execution should not have been terminated after 27 generations, as the average score was still generally decreasing. Time limitations and the proof-of-concept nature of the contract resulted in the decision to terminate computation and move on rather than conducting an exhaustive search for the optimal solution.

The best of generation curve in Figure 7.2 indicates that a superior controller was found on the first generation. This individual had a performance score of 19,844.8 m-s. Subsequent generations produced further improvements in the best of generation individual. Performance scores of 18,829.0 m-s and 17,836.1 m-s were obtained on generations 8 and 24 respectively.

The performance of the genetic algorithms was evaluated by conducting two test flights with several controllers. The selected controllers were the baseline PID autopilot (no neural network), the best of generation individual from generation 0 (with proportional training), and the best of generation (also best of run) individual from generation 24, again with proportional training. The mission profiles used for controller testing are given in Table 7.2 and Table 5.3. Respectively, these are the mission profiles used for genetical algorithm evaluation and off-nominal vehicle performance evaluation.

Table 7.3 lists the gain settings used in each of the tested controllers. An interesting observation is that the autopilot parameters are generally increased over their baseline values by a substantial amount. The exception to this rule is the parameter KH, which is a linear conversion between altitude error and commanded flight path angle. The optimized value of this parameter is seen to be very near to its baseline setting. The primary role of KH is in determining the rate of pitch out as the aircraft approaches its target altitude. Small values of KH produce a slow and lazy acquisition of the target altitude, while excessively large settings create altitude oscillations as the vehicle cannot track the commanded pitch rates. Genetic algorithm results support the conclusion that this parameter is already near-optimal in the existing HATT-X autopilot.

**Table 7.3:** Gain Settings for GA-Controller Evaluations

| Control law variable | Controller | | |
|---|---|---|---|
| | Baseline PID | Generation 0 | Generation 24 |
| K3A | 0.2164 | 1.097 (507 %) | 1.526 (605 %) |
| K3B | -0.226 | -2.256 (998 %) | -1.024 (453 %) |
| KH | 0.0017 | 0.0021 (123 %) | 0.00162 (95 %) |
| KA11A | 4.7467 | 16.955 (357 %) | 11.753 (248 %) |
| KA11B | 0.3745 | 1.101 (294 %) | 2.529 (675 %) |
| $KA11_{MAX}$ | 2.600 | 13.720 (528 %) | 12.901 (496 %) |
| G11 | 0.150 | 0.904 (603 %) | 1.208 (805 %) |
| GACOLI | 5.000 | 15.985 (320 %) | 44.195 (884 %) |

Test flight results for the three controllers are illustrated in Figure 7.3 and Figure 7.4. Examination of Figure 7.3 indicates that the genetically derived controllers assume a much more aggressive attitude in approaching the target altitude. Examination of the raw flight data indicates that these controllers achieve flight path angle on the order of 50°, substantially more than the 15° limit imposed in the current HATT-X autopilot.

Significant overshoot is apparent in the trajectory flown by the "best of generation 0" autopilot. This is particularly noticeable in the final transition to 500 m altitude. Because of the steep dive angle produced, the vehicle airspeed is substantially increased at the bottom of the dive. The control surface authority is insufficient to bring about the large pitch rates required by the control law output, and the vehicle overshoots the target altitude.

After 24 generations, the genetic controller has evolved such that altitude overshoots are significantly reduced. While large acquisition angles are still present, the inner loop controller gains have been adjusted for greater control authority. The overshoot at $t = 93$ *sec* is largely eliminated, while that at $t = 145$ *sec* is reduced by 50 %.

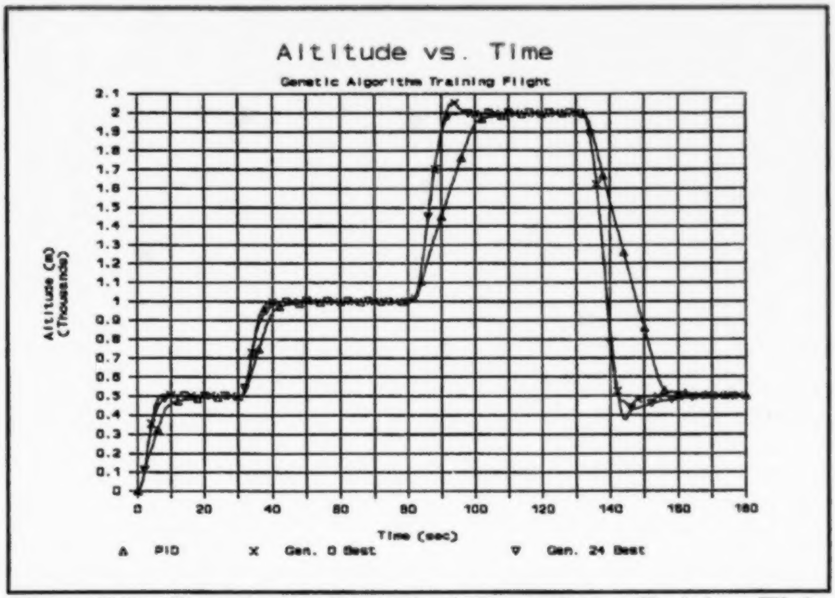

**Figure 7.3:** Genetically Optimized Controller - Training Flight Profile

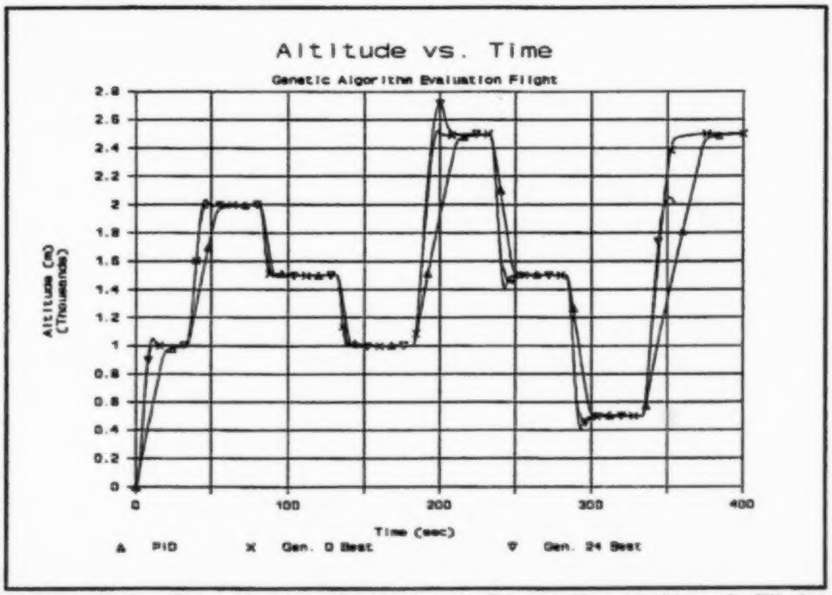

**Figure 7.4:** Genetically Optimized Controller - Novel Flight Profile

In Figure 7.4, the genetic controllers are presented with a novel trajectory to test their generalization ability. As in Figure 7.3, the controllers produce large acquisition attitudes with resulting overshoot of the target altitude. No overall improvement is apparent for the generation 24 individual. In fact, this controller yields inferior results. At $t = 350\ sec$ the aggressive flight path angle bleeds off sufficient airspeed for the vehicle to enter a stall condition. The controller is unable to complete its assigned profile.

This result underscores a point first made in Section 5.3 of the contract interim report[3]. Selection of an appropriate evaluation function is critical to the success of the genetic algorithm. In particular, the evaluation function must suitably represent the full range of conditions likely to be experienced by the controller. In this example, the maximum altitude increase commanded in the training flight was 1000 m. The resulting controller was optimized to conduct transitions of this magnitude. The controller settings are inadequate for a commanded increase of 2000 m, and the vehicle is unable to meet the demands of the mission profile.

### 7.4 Conclusions

Several conclusions were derived from the research conducted in this section. These are itemized below.

1. Genetic algorithms and neural networks may be used in combination to design robust, high performance vehicle controllers. Robust operation is assured by the use of neural networks to adaptively modify immediate controller response, while high performance is assured through the use of genetic algorithms to shape the global characteristics of the controller.

2. The success of optimization via genetic algorithms is highly dependent on the suitability of the evaluation function used. The highly tweaked nature of the resulting controller can cause brittle and potentially unsatisfactory response when subjected to operating conditions outside of its training regime.

3. The requirement to suitably incorporate all possible eventualities in the evaluation function may restrict the applicability of the genetic algorithm technique. This restriction is particularly important in situations where computational power is limited or expensive.

4. The success of the resulting controller is also dependent on the fidelity of the system model used in training and evaluation. For example, the controller generated in this research commands large flight path angles and pitch rates. This control input is acceptable under the assumption of constant thrust. For the actual HATT-X vehicle and its boost-glide thrust characteristics, such a control input would be unacceptable due to the excessive bleedoff of airspeed which results. While this is a gross example, subtle inconsistencies between

the actual system and its software model may also lead to sub-optimal controller settings. In some cases these deficiencies may be nullified by the adaptive nature of the neural network inner loop. In general, however, the potential effects will be unknown and must be carefully examined by the system designer.

## 8.0 Lateral/Longitudinal Neural Network Development

Following completion of the investigation into longitudinal-only flight control via neural networks, Atlantis proceeded to remove the constraint of two-dimensional flight. The following sections document the results obtained after reinstatement of the lateral aerodynamic and control models.

### 8.1 Training Data Generation

Atlantis had intended to conduct training data generation for the combined lateral/longitudinal inverse dynamic model network by extending the approach used for the longitudinal only IDM. However, the simulated aircraft was seen to be highly responsive to aileron inputs. This characteristic necessitated several minor changes to the data generation method.

The first modification required was reduction of the delay between control application and final state measurement from 1.0 to 0.5 seconds. The delay is intended to allow time for the airframe to respond to the control input. It may be considered a characteristic time for the neural network. Reduction of the delay is in keeping with the quicker dynamic response of the controlled system.

Premature termination of data generation test flights was also noted and attributed to the high sensitivity of the vehicle's response to aileron inputs. Stability was added to the vehicle controller by superimposing a rate damping term on the randomly generated control inputs. This modification significantly reduced the tendency of the vehicle to enter high roll angle/roll rate areas of the state space. As a result, the generated training examples more closely reflected typical pilot control inputs. For example, an input state possessing large positive roll or roll rate more frequently produces negative aileron commands. This response produces results similar to the generally expected controller performance, and superior to the blind application of a random control input.

With the exception of the preceding modifications, the data generation method was a direct extension of the procedure documented in Section 5.1 to the dual input case. The data set produced consisted of 7664 training examples, with thirteen inputs and two outputs. This data set was then divided into training and testing data sets with 5000 and 2664 examples respectively. The input/output data structure is defined in Table 8.1.

### 8.2 IDM Network Training

The IDM network training procedure followed was identical to that used in development of the longitudinal-only controller of Section 5.2. In the interests of brevity, the reader is referred to that section for details of the training procedure.

**Table 8.1: I/O Definition for Longitudinal/Lateral IDM Networks**

| Input / Output Name | Description | Range |
|---|---|---|
| INPUTS | | |
| H | Altitude | 0 to 4000 m |
| M | Mach Number | 0.0 to 1.0 |
| $\alpha$ | Angle of Attack | ± 10 ° |
| $\beta$ | Sideslip Angle | ± 5 ° |
| $\theta$ | Pitch Angle | ± 40 ° |
| $\phi$ | Roll Angle | ± 100 ° |
| p | Roll Rate | ± 100 °/s |
| q | Pitch Rate | ± 30 °/s |
| r | Yaw Rate | ± 30 °/s |
| $\delta\text{canard}_0$ | Current canard deflection | ± 12 ° |
| $\delta\text{aileron}_0$ | Current aileron deflection | ± 5 ° |
| $\gamma_{desired}$ | Desired flight path angle | ± 50 ° |
| $\phi_{desired}$ | Desired roll angle | ± 100 ° |
| OUTPUTS | | |
| $\delta$canard | Commanded canard setting | ± 12 ° |
| $\delta$aileron | Commanded aileron setting | ± 5 ° |

As stated in Section 8.1, the input/output data set consisted of thirteen inputs and two outputs. The definition and scaling used for these quantities is described in Table 8.1.

Initial network training runs were geared toward development of a single network which would produce both canard and aileron outputs. These attempts proved unsuccessful due to the large network size required to produce reasonable output accuracy. Subsequent efforts resulted in the development of an individual controller for each output. After comparison of network convergence results, a four-layer network architecture with a 13-15-15-1 structure was selected for use in subsequent investigations.

Interestingly, the IDM networks used to control aileron movements trained to a high degree of accuracy in less than 100 training cycles. In contrast, convergence of the canard control networks was much slower, requiring 2000 cycles on average before converging. The final error level for the canard IDM network was also significantly larger than resulted from aileron IDM training. At the time, it was thought that the large final error was due to an incomplete state definition. All attempts to reduce this error were unsuccessful. The problem source was eventually identified and corrected late in the program schedule. For details of the correction the reader is referred to section 8.5.

## 8.3 Autopilot Integration

Integration of the canard IDM network will the simulated HATT-X autopilot was identical to the longitudinal-only configuration documented in Section 5.3. The operational structure of the autopilot lateral control laws before and after integration of the aileron IDM network are illustrated in Figure 8.1 and Figure 8.2.

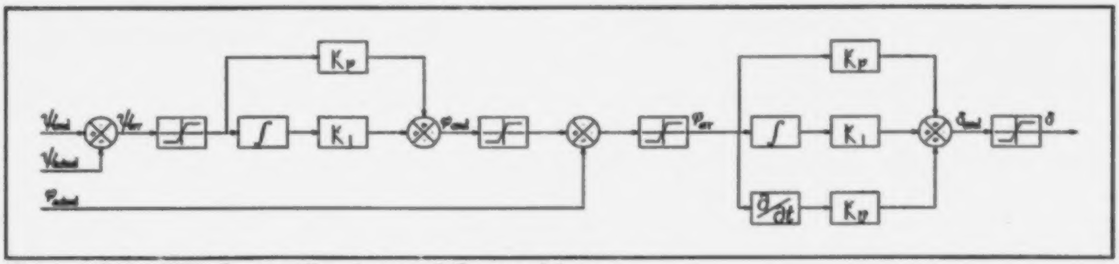
$\psi_{cmd}$
$\psi_{err}$
$\psi_{actual}$
$\varphi_{actual}$
$K_p$
$\int$
$K_I$
$\varphi_{cmd}$
$\varphi_{err}$
$K_p$
$\int$
$K_I$
$d/dt$
$K_D$
$\delta_{cmd}$
$\delta$

**Figure 8.1:** PID Controller Lateral Control Loop

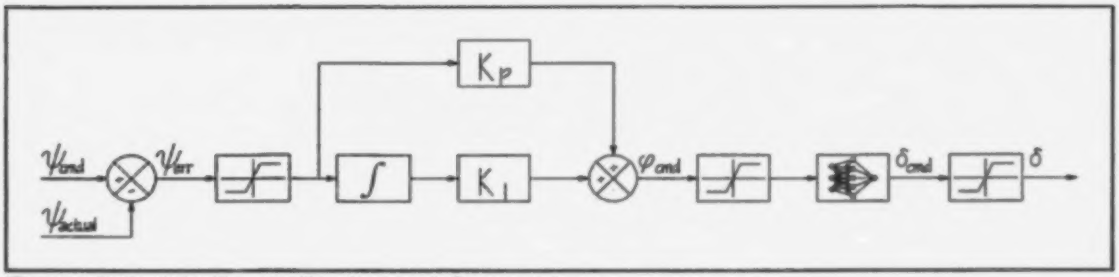
$\psi_{cmd}$
$\psi_{err}$
$\psi_{actual}$
$K_p$
$\int$
$K_I$
$\varphi_{cmd}$
$\delta_{cmd}$
$\delta$

**Figure 8.2:** NN Controller Lateral Control Loop

## 8.4 Test Flight Results

Several test flights were conducted using the combined lateral/longitudinal neural network controller. The mission profiles executed in each flight are described in Table 8.2. Pertinent variable histories from each flight are graphed in Figure 8.3 through Figure 8.17.

**Table 8.2:** Mission Profile Settings for Lateral/Longitudinal Test Flights

| Time (sec) | Test Flight 1 | | Test Flight 2 | | Test Flight 3 | |
|---|---|---|---|---|---|---|
| | Long. mode | Lateral mode | Long. mode | Lateral mode | Long. mode | Lateral mode |
| 0-30 | 1000 m | $\psi = 50°$ | 500 m | $\psi = 50°$ | 1000 m | $\psi = 45°$ |
| 30-80 | 1000 m | $\psi = 90°$ | 2000 m | $\psi = 90°$ | 2000 m | $\psi = 315°$ |
| 80-130 | 2000 m | $\psi = 50°$ | 500 m | $\psi = 180°$ | 500 m | $\psi = 180°$ |
| 130-180 | 3000 m | $\psi = 50°$ | 3000 m | $\phi = -45°$ | 500 m | $\psi = 180°$ |
| 180-230 | 3000 m | $\phi = 30°$ | 500 m | $\phi = 30°$ | 1000 m | $\psi = 315°$ |
| 230-280 | 2000 m | $\phi = -30°$ | 3000 m | $\psi = 0°$ | 2000 m | $\psi = 45°$ |
| 280-330 | 2000 m | $\psi = 90°$ | 500 m | $\phi = -50°$ | 1000 m | $\psi = 45°$ |
| 330-400 | 500 m | $\psi = 270°$ | 500 m | $\phi = 0°$ | 2000 m | $\psi = 315°$ |

The results of the first test flight demonstrate the ability of the neural network controller to track desired flight path and roll angles, and consequently also to capture commanded altitude and heading settings.

It may be noted that on several occasions the vehicle heading diverges from its apparent 0° commanded value. During these periods, the autopilot is in fact executing a roll angle hold manoeuvre, and the vehicle heading is therefore unconstrained. Any ambiguity between heading and roll angle hold modes may be resolved by reference to the MIP descriptions in Table 8.2. In addition, roll angle hold is distinguished by the presence of a constant roll angle command, while the commanded roll angle varies continuously in heading hold mode.

### 8.4.1 Lateral Test Flight 1

Results for the first test flight are generally acceptable. In virtually all cases, the IDM controller is able to track the desired values of roll, heading, and altitude. Control surface outputs are generally smooth. However, small oscillations occur after changes in the desired control setting. These are quickly damped as the controller adjusts to its new operation regime.

Some errors are apparent in the graph of desired and actual roll angles. The neural controller consistently overshoots when presented with a step variation in commanded attitude. Overshoots of up to 5° are observed. With one exception, the roll angle quickly subsides to its commanded value. The sole exception is the transition to 30° commanded at t=180 seconds. For this case the vehicle overshoots and does not return, yielding a steady state error of approximately 2.7°.

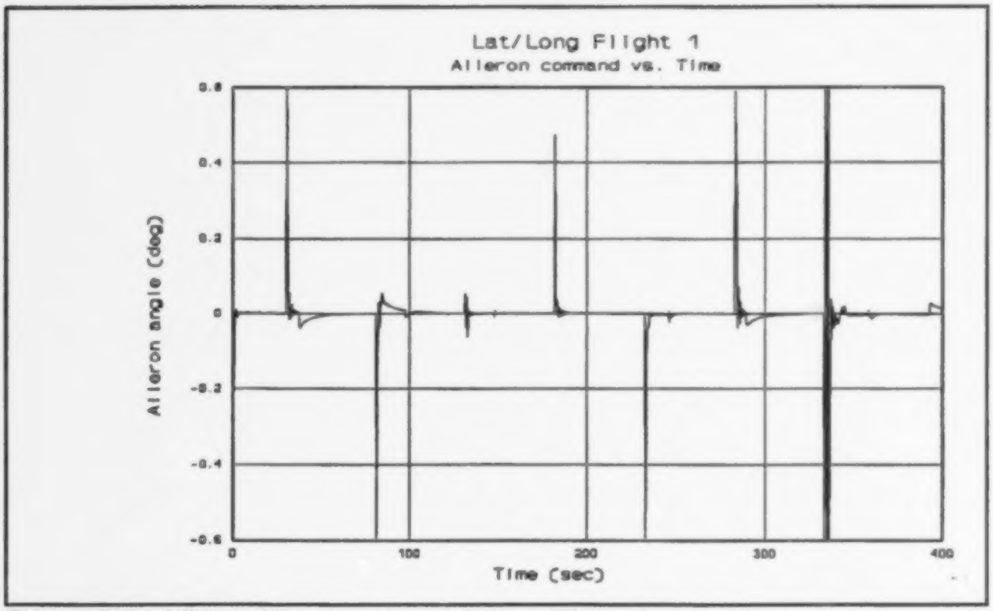

Figure 8.3: Aileron History - Lateral Test Flight 1

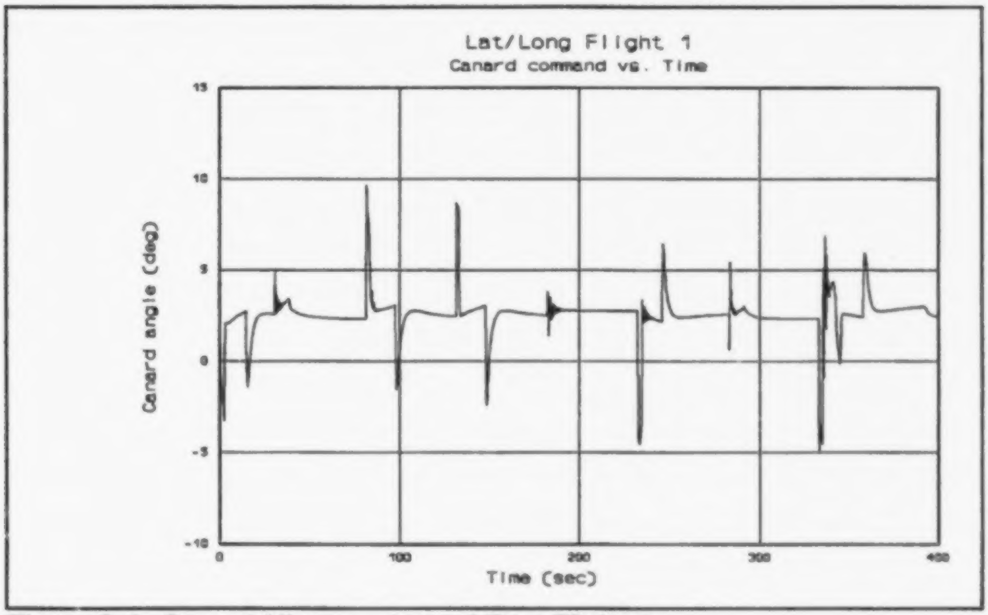

**Figure 8.4:** Canard History - Lateral Test Flight 1

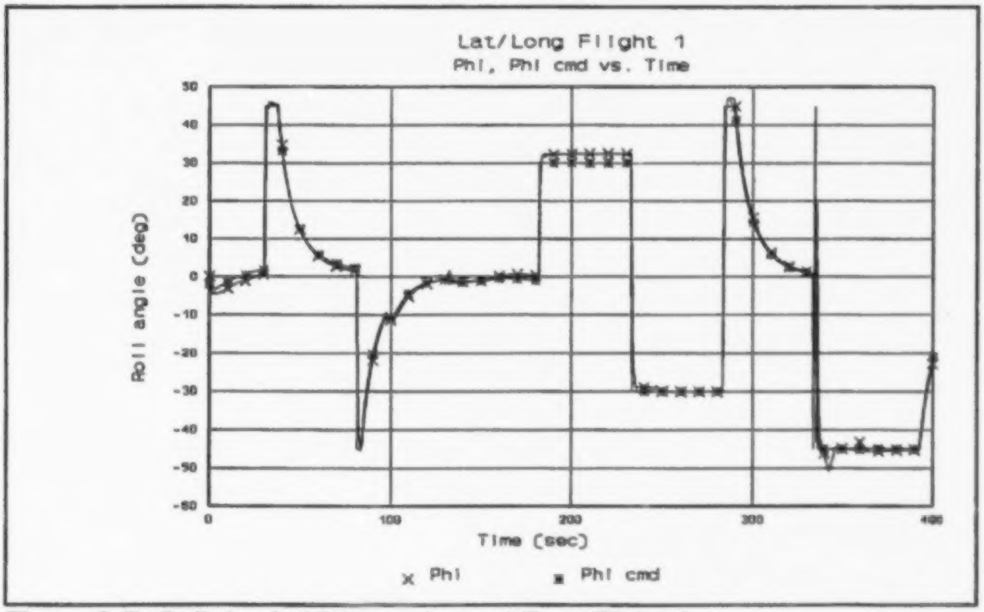

**Figure 8.5:** Roll Angle History - Lateral Test Flight 1

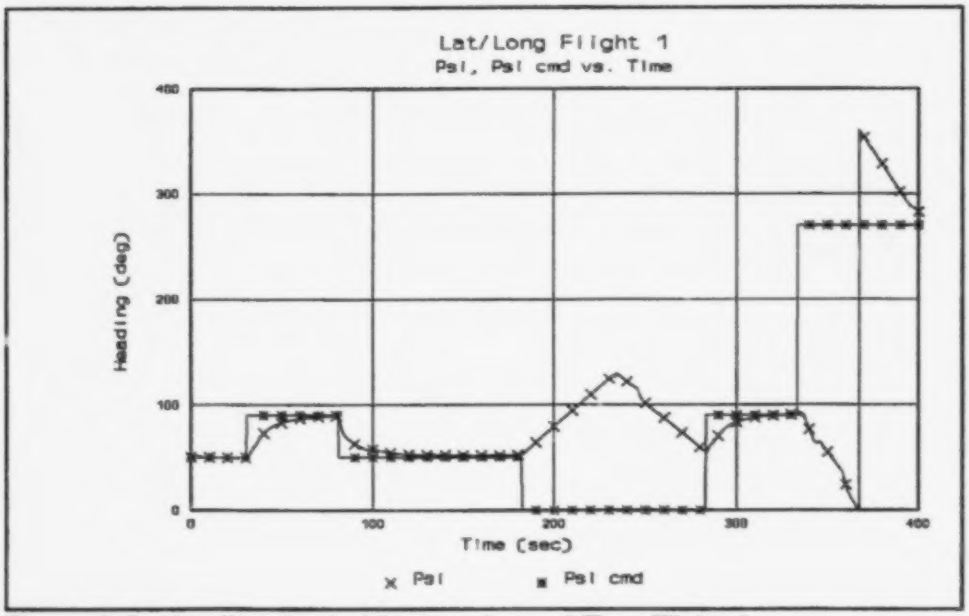

**Figure 8.6:** Heading Angle History - Lateral Test Flight 1

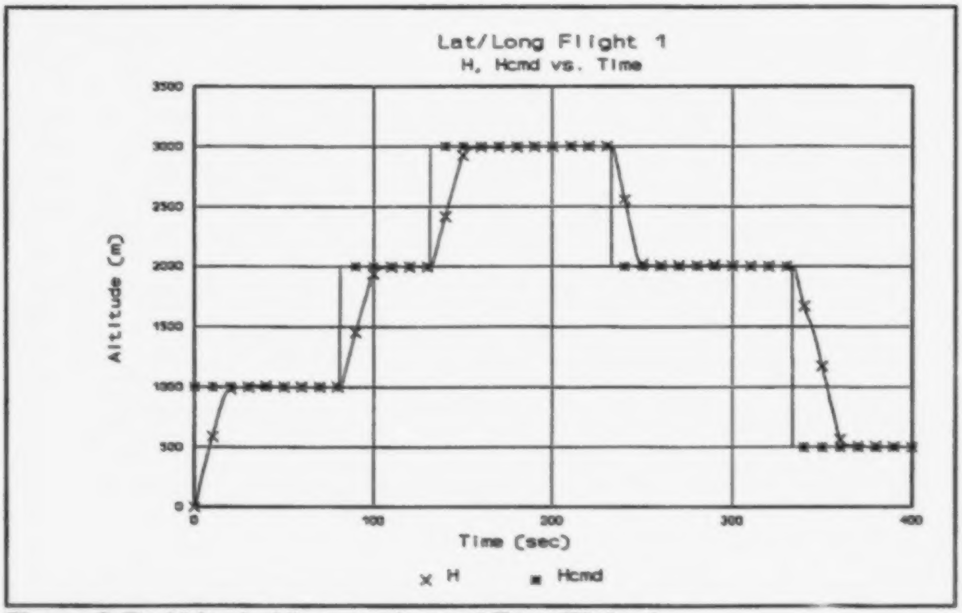

**Figure 8.7:** Altitude History - Lateral Test Flight 1

### 8.4.2 Lateral Test Flight 2

Examination of the variable histories from the second test flight shows a sudden onset of oscillation in the controller outputs after the commanded state transition at t = 280 seconds. Substantial effort was expended in an attempt to identify and resolve the cause of this oscillation. This work is documented in Section 8.5, and will not be discussed here.

Note that vehicle controllability is maintained (albeit at a greatly reduced performance level) during the period of oscillation. This fact is largely due to the damping effect of the first order lag introduced by the servo dynamic model.

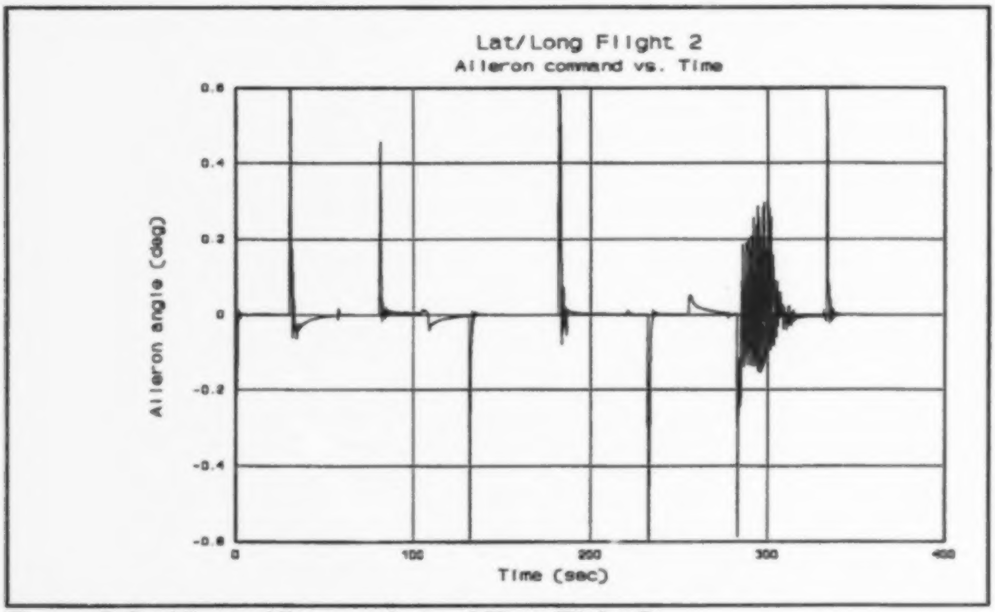

Figure 8.8: Aileron History - Lateral Test Flight 2

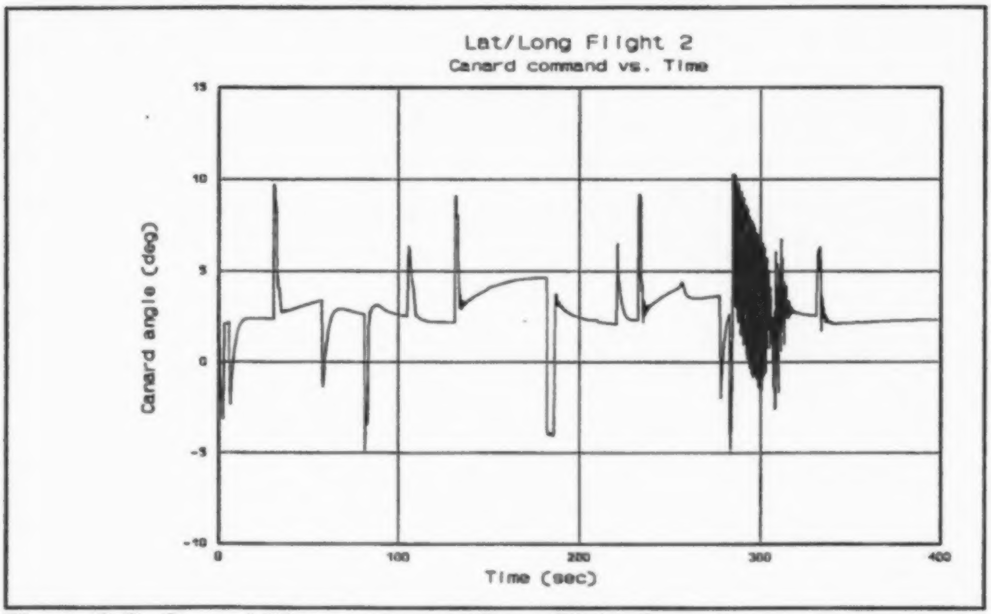

**Figure 8.9:** Canard History - Lateral Test Flight 2

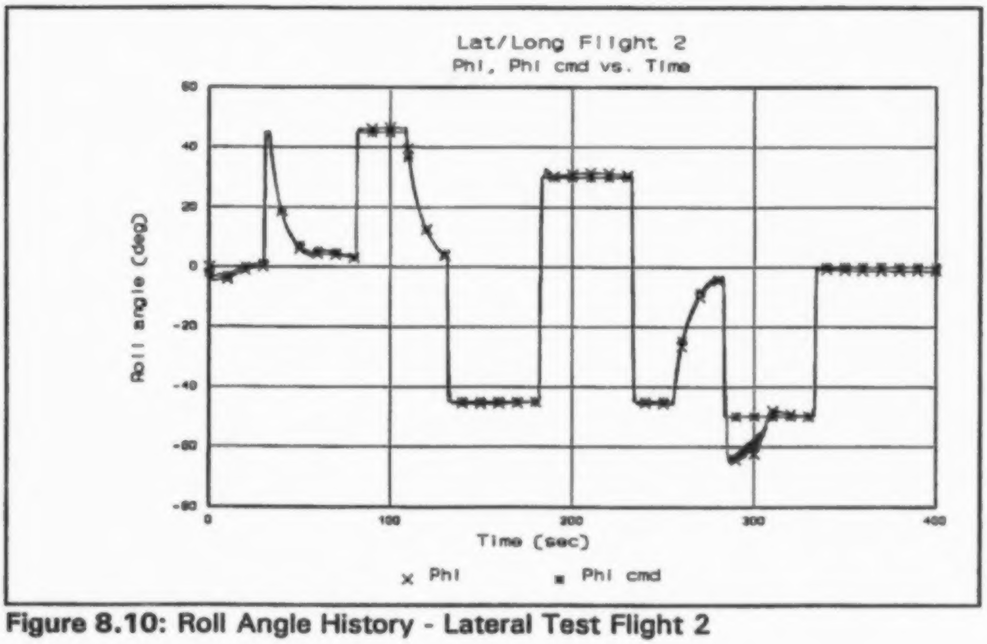

**Figure 8.10:** Roll Angle History - Lateral Test Flight 2

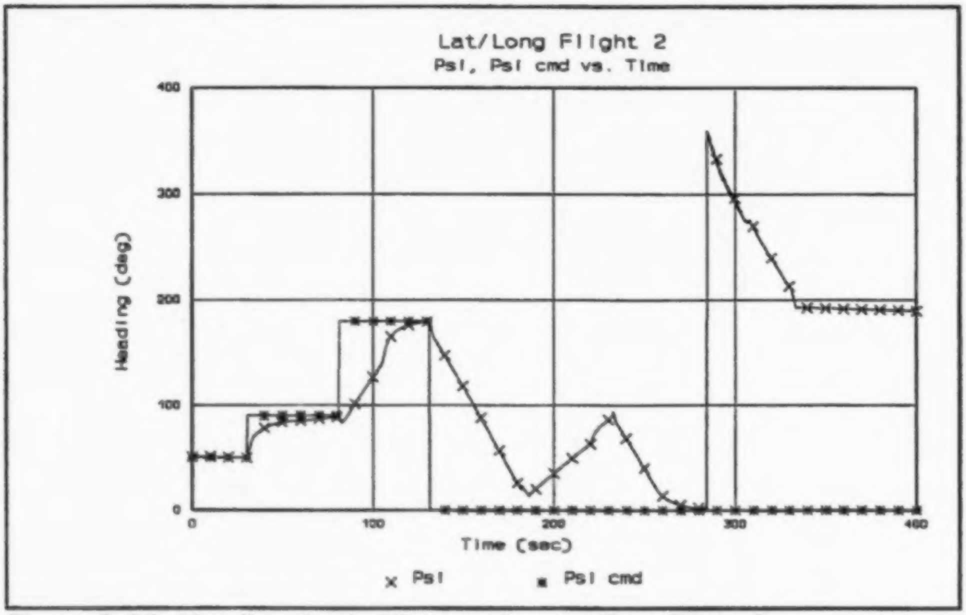

**Figure 8.11:** Heading Angle History - Lateral Test Flight 2

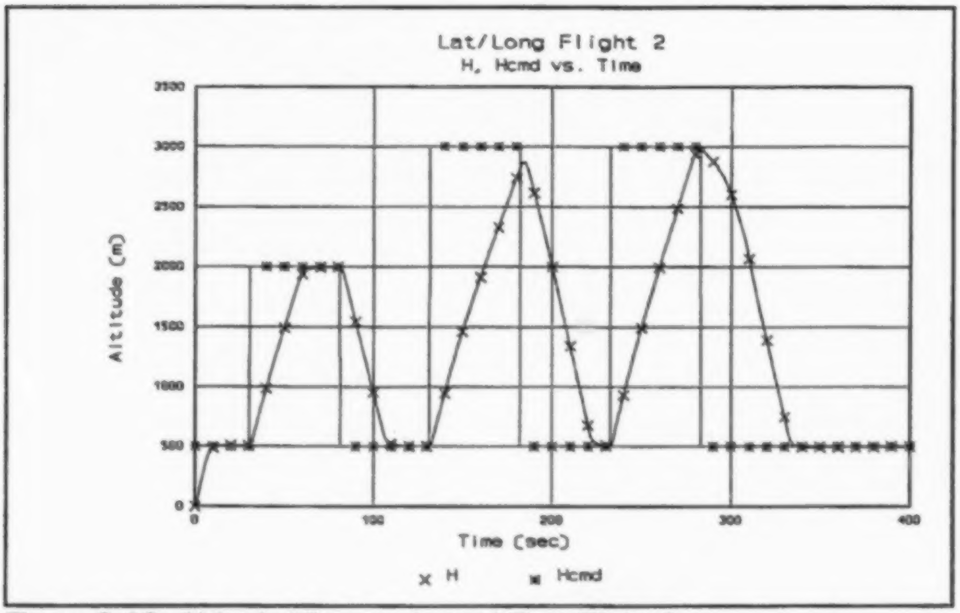

**Figure 8.12:** Altitude History - Lateral Test Flight 2

### 8.4.3 Lateral Test Flight 3

Canard surface oscillations appear in this flight after the 330 second command transition. The severity of the oscillation is significantly less than that encountered during the second test flight. Small instabilities are also apparent on the aileron channel. However, the vehicle is able to track the commanded roll angle with very good accuracy.

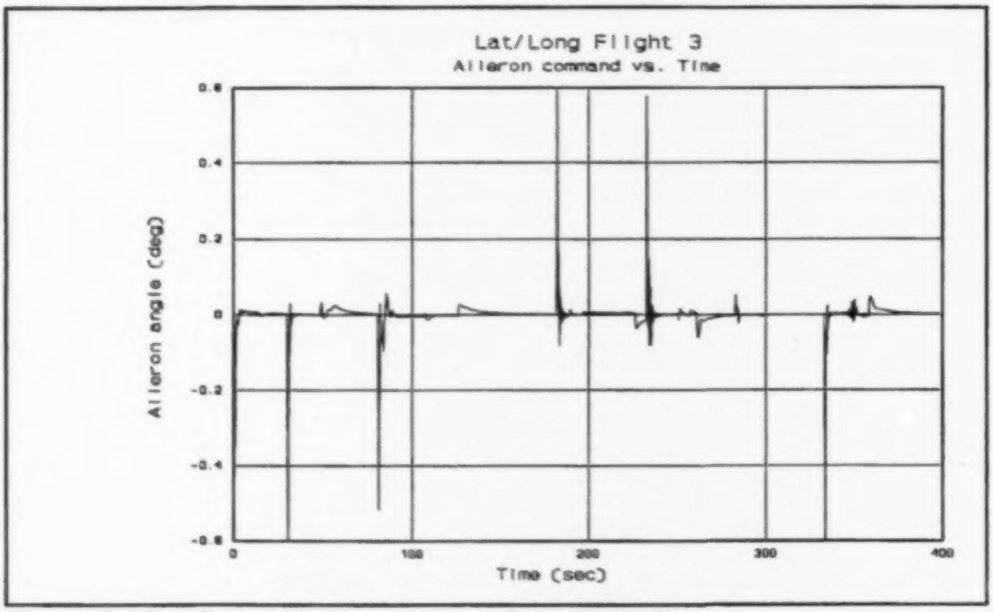

**Figure 8.13:** Aileron History - Lateral Test Flight 3

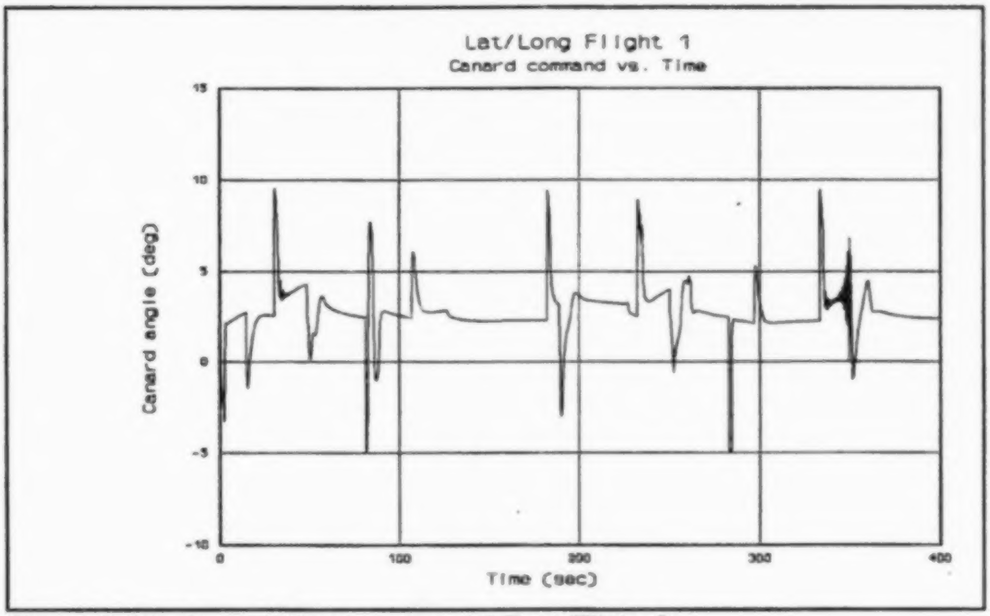

**Figure 8.14:** Canard History - Lateral Test Flight 3

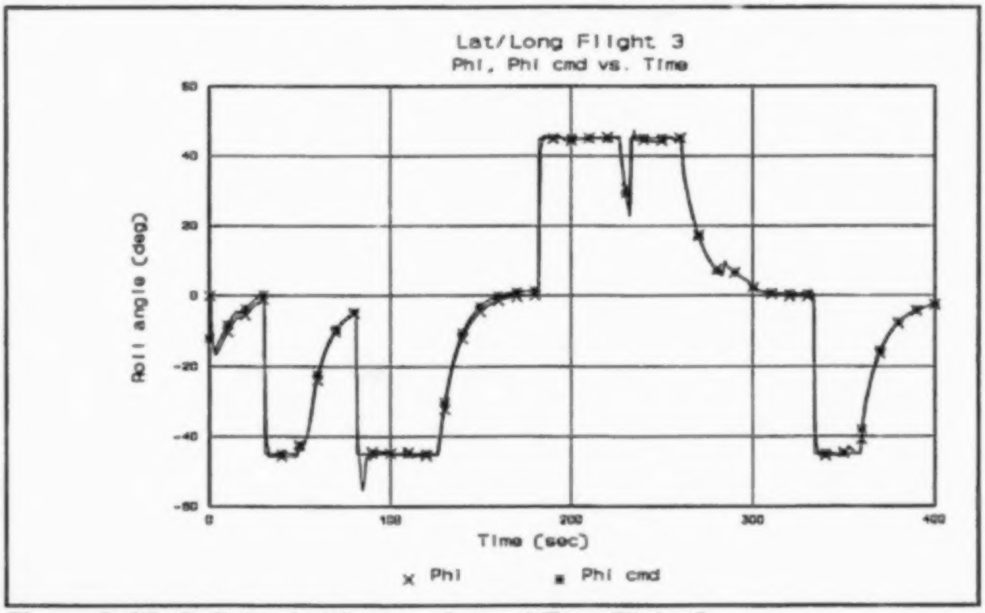

**Figure 8.15:** Roll Angle History - Lateral Test Flight 3

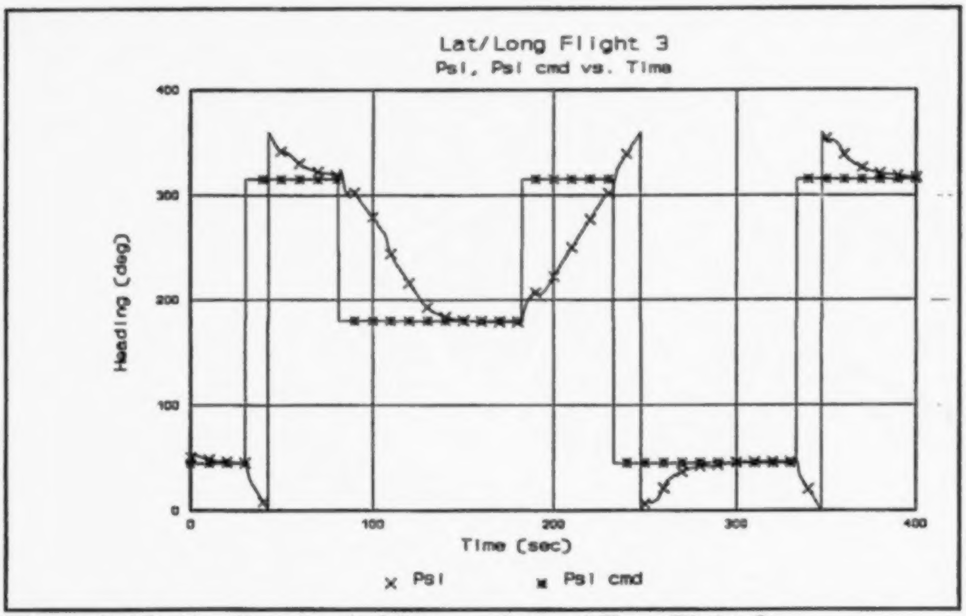

**Figure 8.16**: Heading Angle History - Lateral Test Flight 3

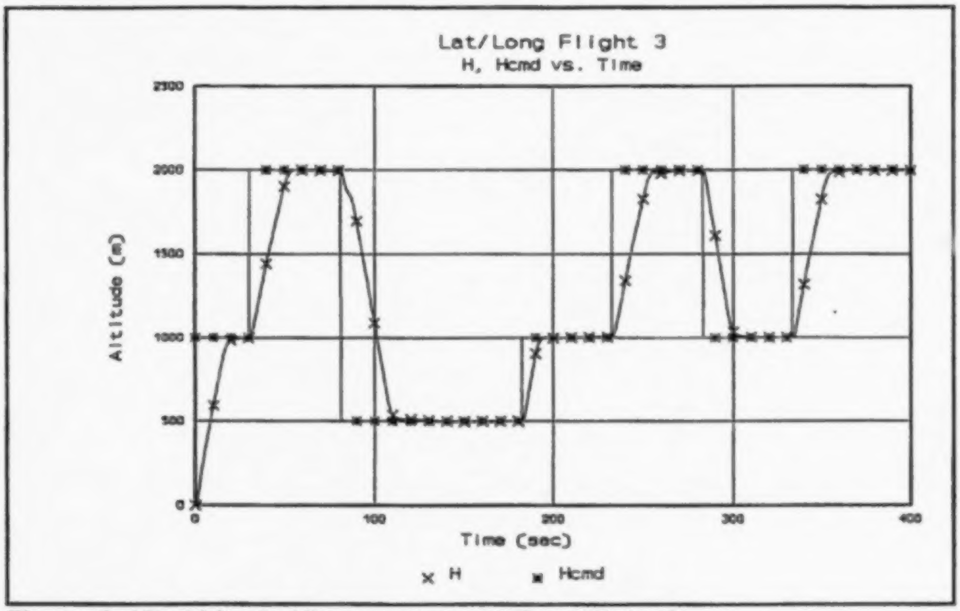

**Figure 8.17**: Altitude History - Lateral Test Flight 3

### 8.5 Network Stability Analysis

The flight test results of Section 8.4 demonstrated that unwanted oscillations can develop in the IDM network outputs. These oscillations degrade vehicle controllability with potentially catastrophic results. Flight safety is therefore highly dependent on ensuring that such oscillations are not allowed to develop. Accordingly, substantial efforts were made to characterize the source of these oscillations and to develop methods for their suppression.

Examination of the flight test results showed that oscillations develop after each change in the vehicle control command. Associated with each of these transitions is a shift in the IDM network input vector to a region of comparatively larger output error. Some small oscillations are expected as the network reduces the local error in its new operation regime. These minor oscillations are readily apparent in the IDM output plots and generally subside quickly. They are a normal artifact of the network training algorithm and have little impact on vehicle performance.

In some cases, however, the oscillation does not subside but rapidly increases in magnitude. Atlantis has presented the hypothesis that onset of sustained oscillation results from training in a region of high sensitivity to local weight changes. More specifically, the network output in the new flight regime is very sensitive to changes in the network weights. Weight adjustment by backpropagation of error causes rapid shifts in the network output, which quickly build up to produce the large scale oscillation observed.

Initial investigations were focused on testing this hypothesis. A generally observed phenomenon is that network weight oscillations are suppressed by a reduction in the network learn rate. However, a low learn rate reduces the network's ability to adjust to new operational conditions. An appropriate compromise must therefore be found between smoothness of weight updates and speed of learning.

A tradeoff between these two conflicting goals was obtained by implementing an algorithm to dynamically adjust the network learn rate. The slew rate of the network output was monitored and used to vary the network learn rate according to the following equations:

$$\mu_{t+\delta t}=\left\{\begin{matrix}0.99\mu_t+0.01\\ \\ 0.99\mu_t\end{matrix}\right\}\text{ if }\left\{\begin{matrix}\dfrac{o_{t+\delta t}-o_t}{\delta_t}>\zeta\\ \\ \dfrac{o_{t+\delta t}-o_t}{\delta_t}<\zeta\end{matrix}\right\} \tag{8.1}$$

$$\eta_t=\eta_0(1-\mu_t) \tag{8.2}$$

where

| | |
|---|---|
| $t$ | is a time index time |
| $\delta t$ | is the autopilot update interval |
| $o_t$ | is the network output at time $t$ |
| $\zeta$ | is a characteristic output slew rate |
| $\mu_t$ | is a non-dimensional parameter indicating degree of output instability at time $t$ |
| $\eta_0$ | is the default network learn rate |
| $\eta_t$ | is the network learn rate at time $t$ |

$\mu_t$ indicates the degree of stability of the network output over the last few seconds of flight. If the network output is changing rapidly (as occurs during periods of oscillation), $\mu_t$ tends to increase. When the network output is relatively constant, $\mu_t$ decays to zero. The parameter $\zeta$ provides a definition of "changing rapidly." $\zeta$ values of 0.2 °/second and 2.0 °/second were used for the aileron and canard networks respectively.

On each iteration of the autopilot logic, the network learn rate is reduced from its default value of $\eta_0$ by an amount proportional to $\mu_t$. By reducing the learn rate during periods of large output changes, it is hoped that large weight changes and corresponding output oscillations will be reduced.

It should be noted that learn rate reduction during periods of large output swings can be justified even if weight oscillation is not an issue. A fundamental assumption of the IDM network training algorithm is that the control surface setting remains unchanged during the period from initial command to final state measurement. This assumption is clearly violated if large output slew rates are present. It is therefore logical to reduce network learning during such periods.

The modifications presented were tested on several simulated flights. All tests were conducted using the mission profile from test flight 2. These runs were used to determine appropriate settings for the parameters and to evaluate the algorithm's impact on vehicle performance. Controller performance was not found to improve significantly on any of the runs. Although the output oscillations were slightly reduced, a significant amount of instability remained. In addition, the reduced learn rate increased the network's reaction time, resulting in large control lags and overshoots in transitions to new commanded states.

Given the apparent failure of this hypothesis, Atlantis instigated a detailed review of all neural network inputs, outputs, and operations. At this time it was discovered that the network input set consisted of a combination of raw and filtered state variables. To improve consistency, the network input vector was corrected to use filtered quantities only. In all cases where more than one filtered value was available, the quantity used in the MPS-100 control laws was selected for input to the network.

IDM network training was repeated using the revised input-output mapping. As was mentioned in Section 8.2, training of the original canard IDM network was considerably more difficult than aileron network training. In contrast, both aileron and cnaard IDM networks trained to a low error level in less than 200 cycles when the modified training set was used. The improved training performance may be considered confirmation that the modified input-output mapping is "better" than the original network configuration.

To test the performance of the revised network mapping, test flight 2 from Section 8.4 was repeated using the newly trained IDM networks. Variable profiles from this flight are included in Figure 8.18 through Figure 8.22. These results show a clear improvement in the IDM network stability characteristics. Small oscillations are apparent after each control mode change, however these are an expected byproduct of the network training method used. The oscillations are of a minor nature and quickly die away to zero as the network refines its output mapping in the vicinity of the new operating point.

It is interesting to note that the modified network tends to produce a larger overshoot in the capture of a new roll angle command. This effect is probably due to the use of low pass filtered (and therefore lagged) variables in the network input. Whatever the source, the controller quickly acts to correct the overshoot. The use of lagged inputs is not anticipated to cause any significant control difficulties. However, this is an item of potential concern and should be monitored in the course of subsequent research.

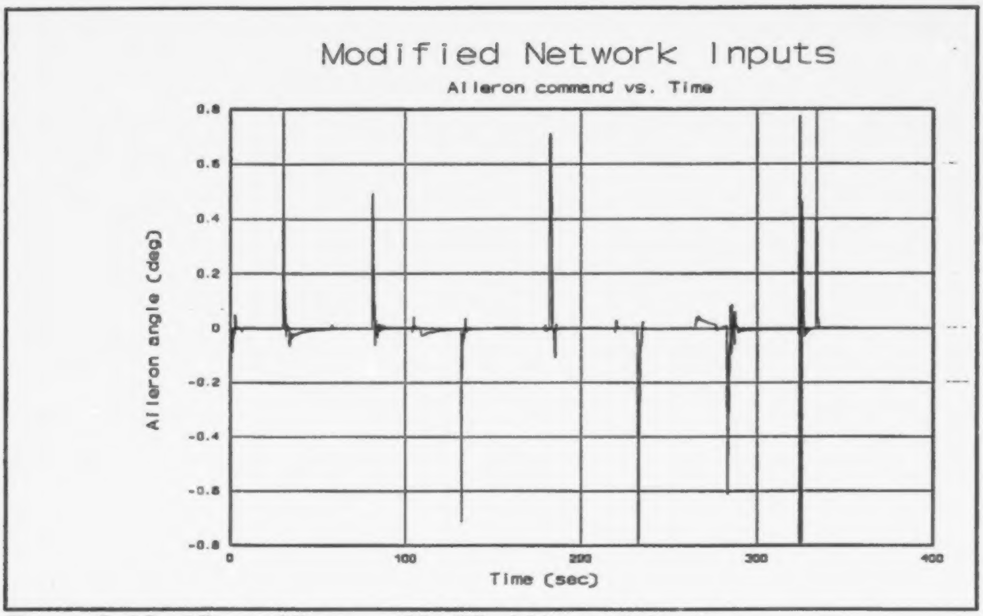

**Figure 8.18:** Aileron History - Modified Network Inputs

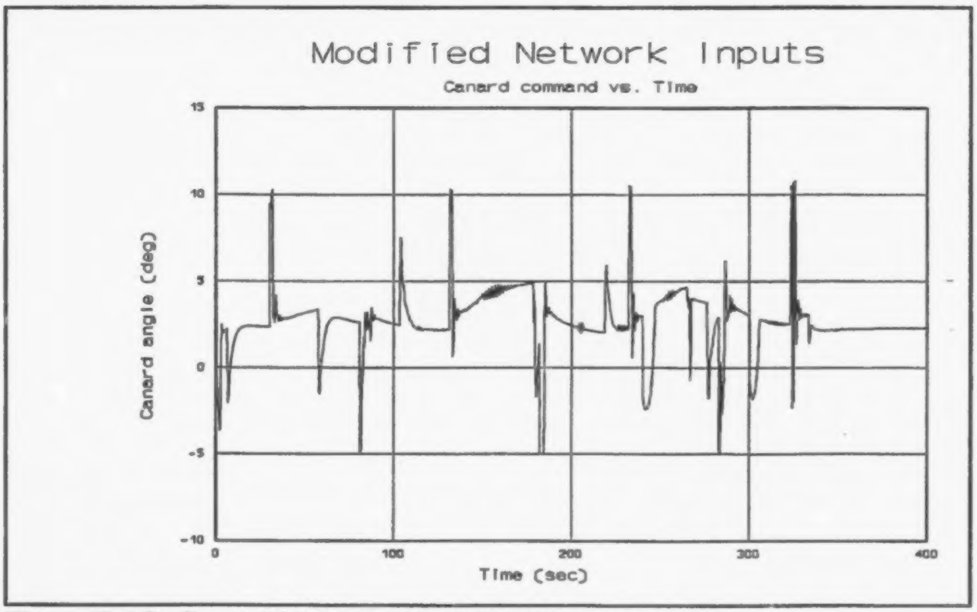

**Figure 8.19:** Canard History - Modified Network Inputs

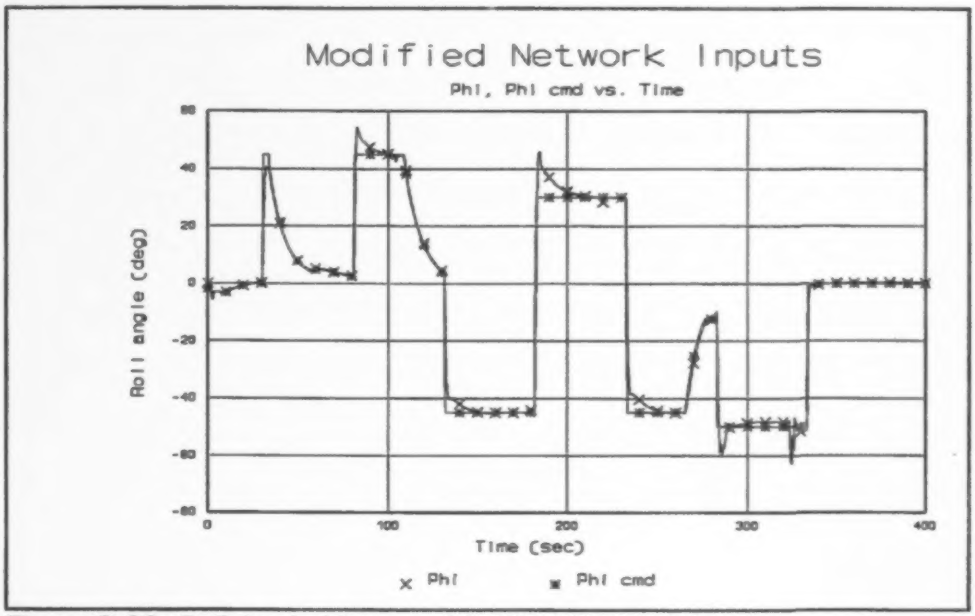

Figure 8.20: Roll Angle History - Modified Network Inputs

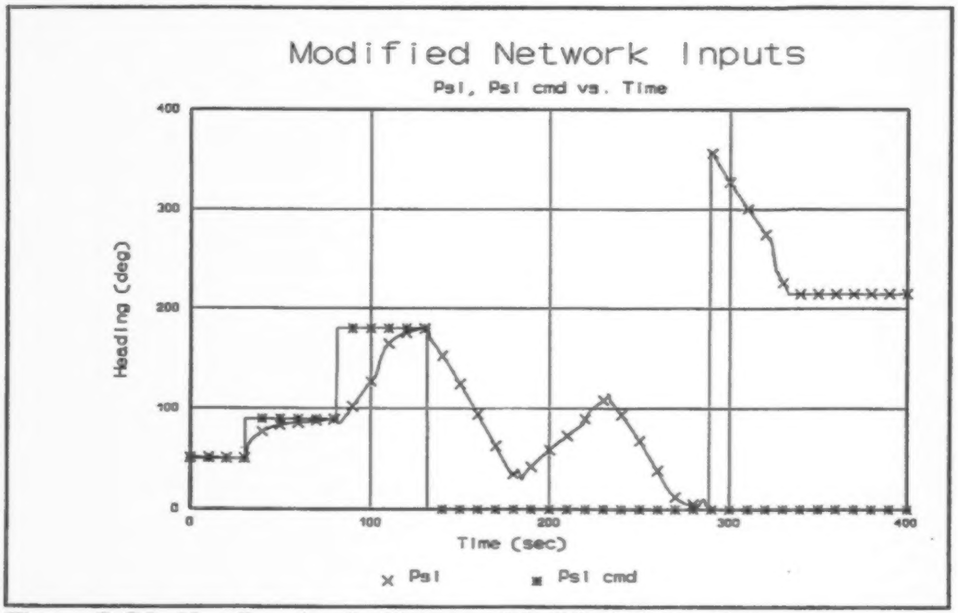

Figure 8.21: Heading Angle History - Modified Network Inputs

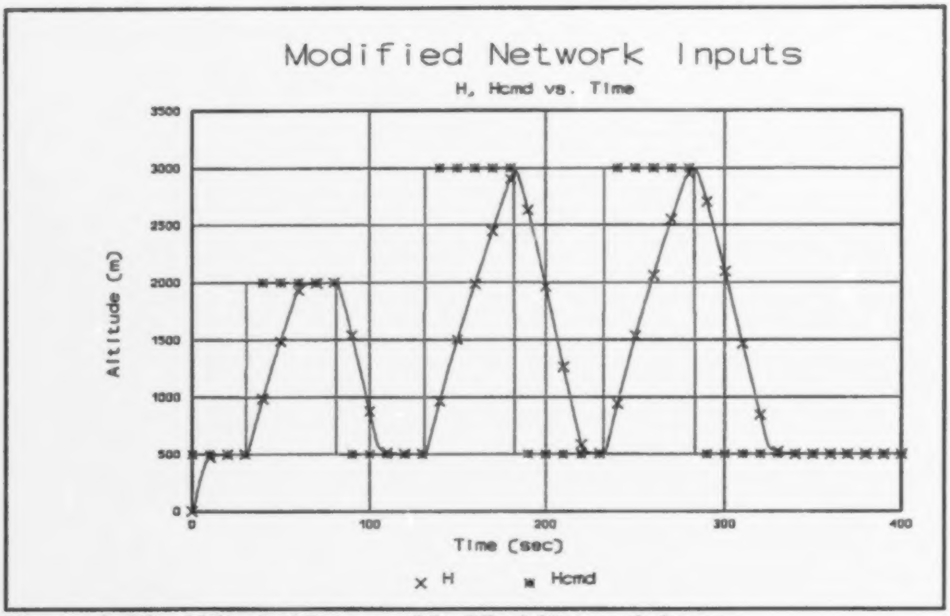

**Figure 8.22:** Altitude History - Modified Network Inputs

## 9.0 Application to Off-Nominal Conditions

### 9.1 Controller Sensitivity to Sensor Noise

A potential area of concern in the application of a neural network controller to an actual HATT-X flight will be the effect of noisy sensor inputs on controller performance. It is possible that excessive levels of sensor noise may disrupt the IDM network mapping, resulting in inaccurate control outputs.

The effects of sensor noise were investigated by adding a random noise component to the roll, pitch, and yaw gyro values used in autopilot calculations. Noise contributions were recalculated once per iteration through the autopilot control loop.

For comparative purposes, the IDM network and PID controllers were each flown through a mission profile with the rate gyro noise components active. Noise values were normally distributed with zero mean and standard deviations of three and ten degrees/second on the first and second runs respectively. Identical noise profiles were applied to each controller. The mission profile used consisted of the following sections:

a. Altitude hold at 500 m; roll angle hold at 0° bank angle.
b. Altitude hold at 500 m; roll angle hold at 30° bank angle.
c. Altitude hold at 1000 m; roll angle hold at -30° bank angle.
d. Altitude hold at 1000 m; roll angle hold at 0° bank angle.

Altitude and bank angle profiles for each controller are provided in Figure 9.1 through Figure 9.4. The performance of the neural network variant is seen to be roughly equivalent to the PID controller in both longitudinal and lateral control.

These results indicate that noise resistance is not improved using the neural controller. However, they do serve as preliminary confirmation that the IDM network approach is not degraded by operation in an uncertain environment.

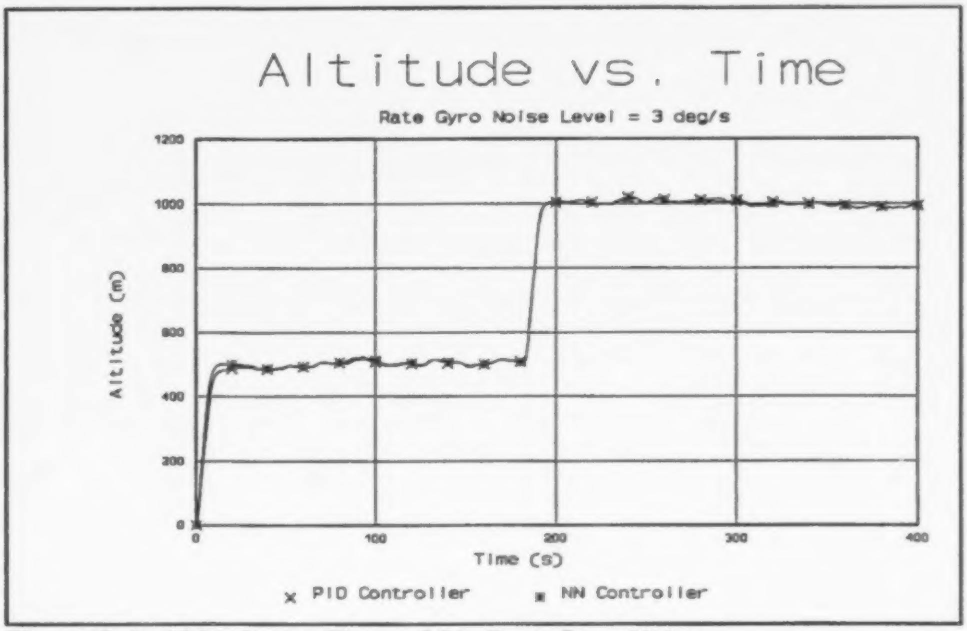

**Figure 9.1:** Altitude vs. Time - 3°/s Rate Gyro Noise

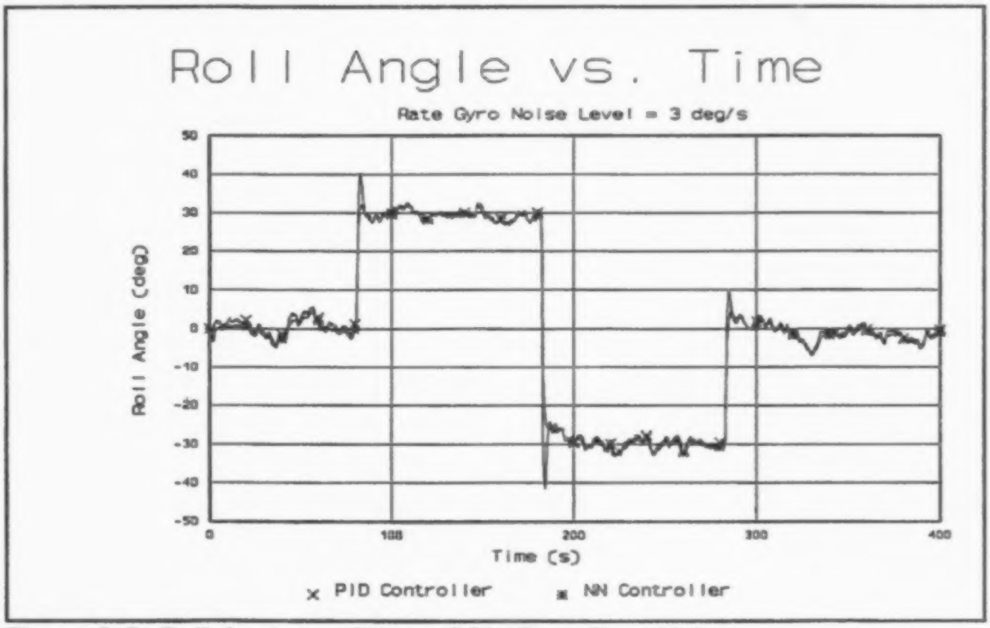

**Figure 9.2:** Roll Angle vs. Time - 3°/s Rate Gyro Noise

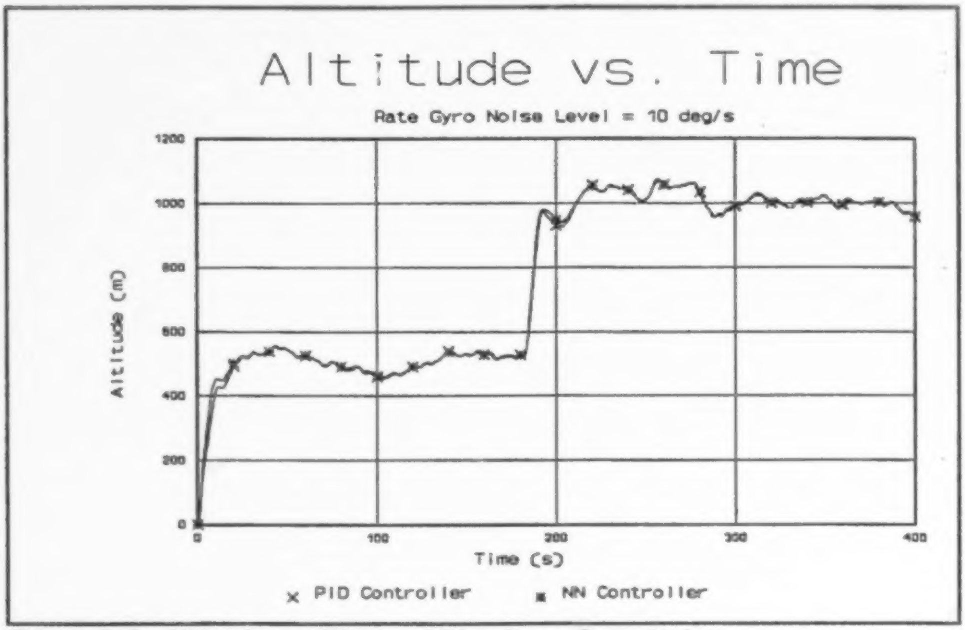

**Figure 9.3: Altitude vs. Time - 10°/s Rate Gyro Noise**

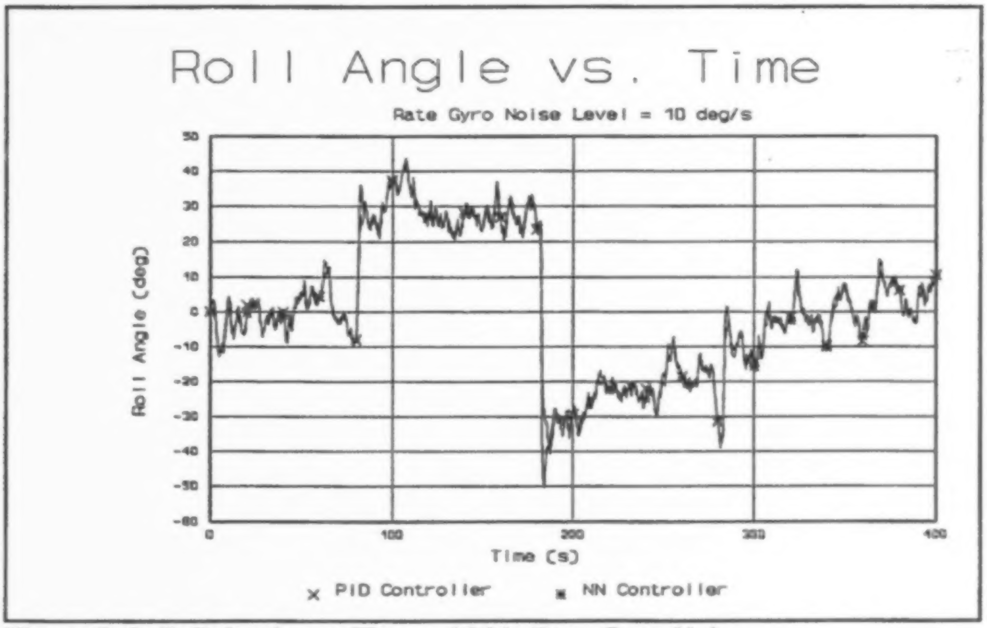

**Figure 9.4: Roll Angle vs. Time - 10°/s Rate Gyro Noise**

### 9.2 Increased Servo Lag

The FLISIM simulation package models the dynamic characteristics of the HATT-X servomotors by implementing a first order lag between commanded and actual servo positions. The adaptability of the neural controller was further evaluated by increasing the characteristic lag time of the aileron/canard servos by a factor of 25 %. The effect of this modification is to create a more "sluggish" servo. To compensate for the change, a controller must select between two options. First, it can anticipate required actions and issue commands earlier, thereby allowing more time for the servo to achieve the commanded setting. Second, it can command a larger deflection, thereby forcing the servo to slew toward the correct position at a higher rate.

The neural network's ability to adapt to this system degradation was assessed through comparison with the baseline PID controller's performance under identical conditions. Variable histories for each controller are illustrated in Figure 9.5 through Figure 9.7. The mission profile used for these flights was that of lateral/longitudinal test flight 2.

The test results of Figure 9.5 indicate that the neural network has little effect on the altitude stability of the simulated vehicle. The PID and neural controllers both exhibit significant longitudinal performance degradations when applied to this plant variant.

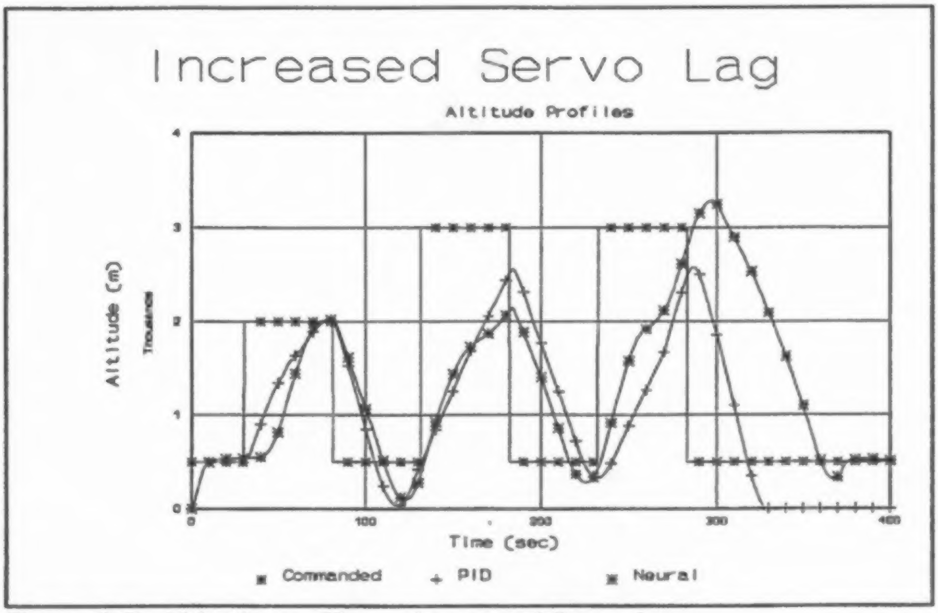

Figure 9.5: Altitude vs. Time - Increased Servo Lag

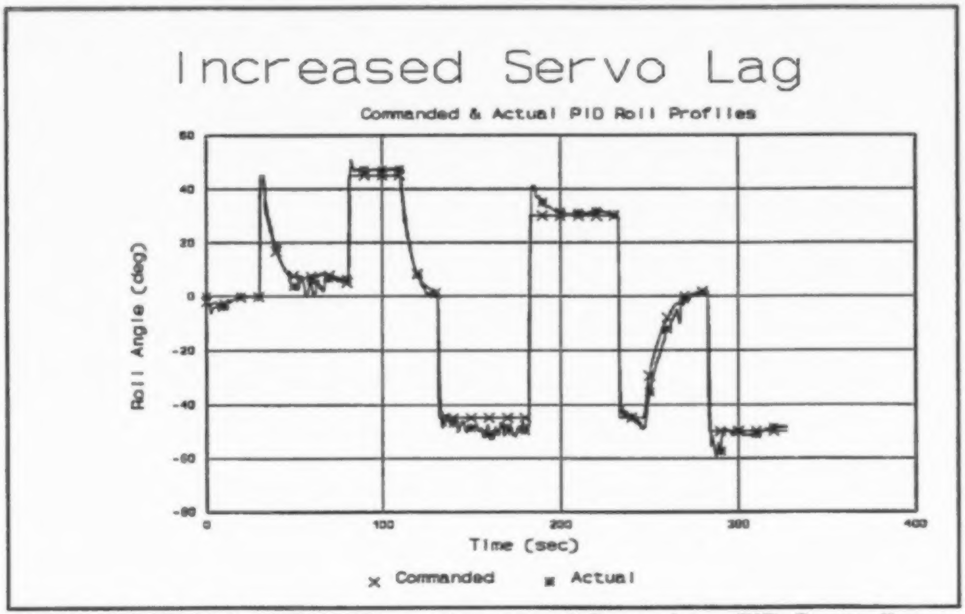

Figure 9.6: Roll Angle vs. Time - Increased Servo Lag, PID Controller

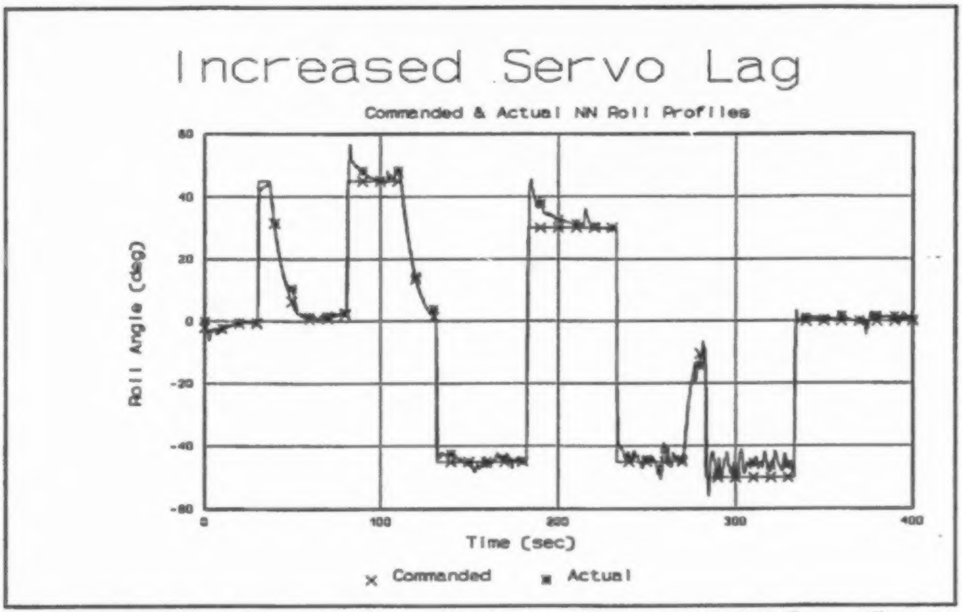

Figure 9.7: Roll Angle vs. Time - Increased Servo Lag, NN Controller

Figure 9.6 and Figure 9.7 respectively illustrate the roll angle capture and track capabilities of the PID and neural controllers. Again, the neural network controller appears to have little impact on vehicle performance. Substantial overshoot and oscillation is apparent in each case.

The poor performance of the neural network on this test case is somewhat surprising, given the rapid learning and adjustment capabilities which have been exhibited on other off-nominal flights. Performance may depend on the ratio between the servo response time and the network's control application-measurement lag time. A ratio of approximately one implies that the new control setting can have little effect on the vehicle trajectory prior to final state measurement. A ratio significantly less than one allows the servo and subsequently the vehicle to respond to the new command setting before the final state is read. The impact of the control setting is more apparent to the network, thereby allowing easier training.

### 9.3 Constant Servo Error

In this test, constant offsets of 0.5° and 2.0° were added to the commanded aileron and canard positions respectively. The effect of this test is to simulate a degraded mode of operation, possibly arising from errors in vehicle assembly.

Altitude and roll angle results for this test are illustrated in Figure 9.8 through Figure 9.10. Examination of Figure 9.8 indicates little difference in the controllers' ability to track and capture a commanded altitude. The neural controller does achieve a more agressive climb angle, however the difference between the two controllers is slight.

In contrast, lateral test results show that the IDM network controller provides a clear advantage over the traditional PID control laws when applied to this test case. As expected, the IDM network displays an initial error which is quickly reduced to zero following each control transition. The PID controller does not perform nearly as well, exhibiting a near-uniform roll error of approximately twenty degrees throughout the duration of the flight.

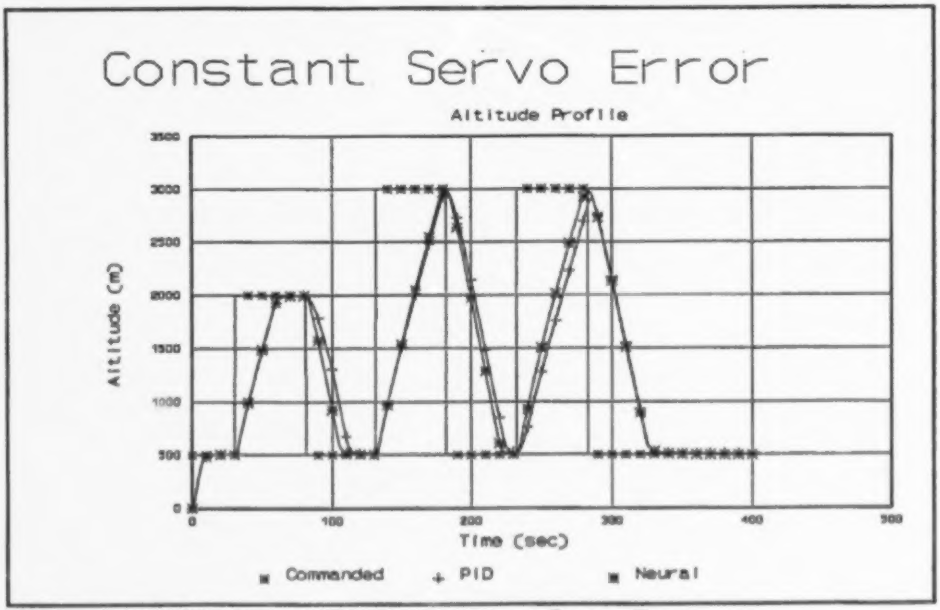

Figure 9.8: Altitude vs. Time - Constant Servo Error

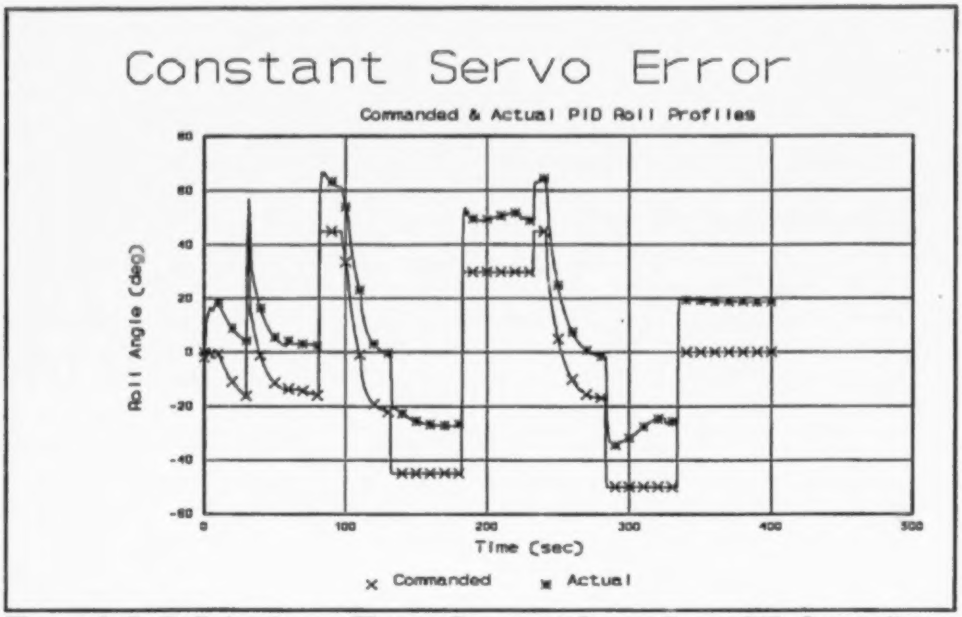

Figure 9.9: Roll Angle vs. Time - Constant Servo Error, PID Controller

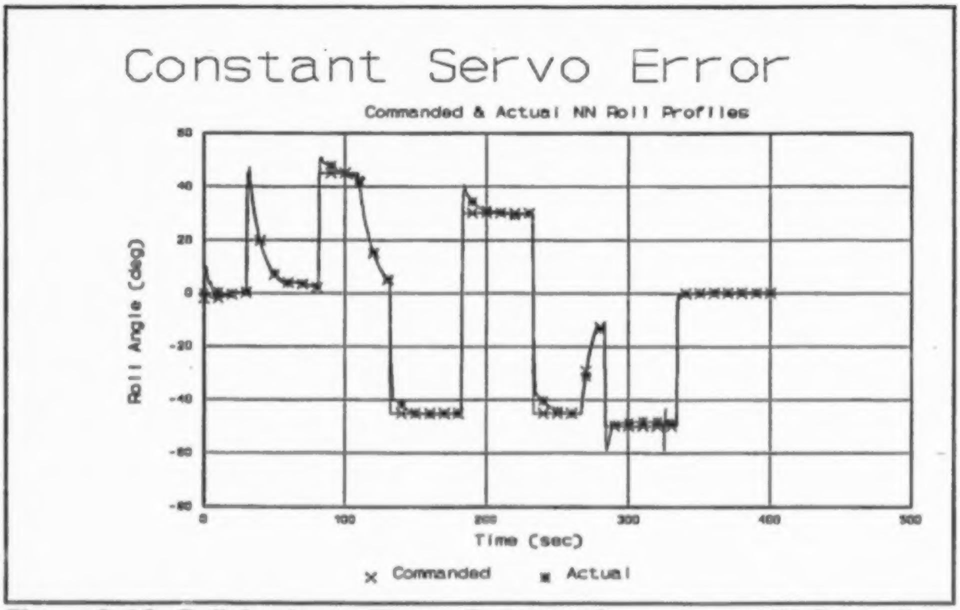

Figure 9.10: Roll Angle vs. Time - Constant Servo Error, NN Controller

The apparently constant nature of the PID controller's roll error is surprising, as the controller's integral term should increase with time to null the servo positioning error. This does not appear to be the case. The MPS-100 autopilot does limit the integral term's contribution to the total servo output. This was immediately thought to be the cause of the persistent roll error. However, a check of the simulation source code revealed that the integral contribution is limited to ±4°. This range should allow the controller to counteract the 0.5° positioning error applied to the roll servos.

## 10.0 Application of Genetic Algorithms to Lateral/Longitudinal Controller Optimization

### 10.1 Test Configuration

Genetic algorithms were applied to optimization of the longitudinal and lateral controller outer loops. The existing optimization procedure was extended to encompass the necessary parameters for three-dimensional flight. Longitudinal control law parameters optimized by the genetic algorithm were those used in the two-dimensional run, and may be found in Table 7.1. The additional lateral control law parameters and their baseline settings are itemized in Table 10.1. Each generation consisted of 200 individuals, with each individual evaluated over an identical flight profile of 200 seconds duration. Each controller was evaluated using the performance function

$$V=\int_{t=0}^{200}\left(K_h n_h(t)\,|h_c(t)-h(t)|+K_\psi n_\psi(t)\,|\psi_c(t)-\psi(t)|\right)dt \qquad \textbf{(10.1)}$$

where

| | |
|---|---|
| $h_c(t)$ | is the commanded altititude at time t. |
| $h(t)$ | is the actual altitude at time t. |
| $\psi_c(t)$ | is the commanded heading at time t. |
| $\psi(t)$ | is the actual heading at time t. |
| $K_h$ | is a relative weighting applied to altitude errors. |
| $K_\psi$ | is a relative weighting applied to heading errors. |
| $n_h(t)$ | is set to 1 whenever a new altitude command is issued, and increases by 1 each time the vehicle overshoots the target altitude. |
| $n_\psi(t)$ | behaves identically to $n_h(t)$, but is based on commanded and actual heading. |

Results from the longitudinal-only development indicated that the genetic algorithm selected excessively steep climb and dive angles in order to minimize its cost function. As a result, large overshoots were apparent in the flight trajectories. The introduction of $n_h$ and $n_\psi$ is a direct attempt to reduce this tendency. These terms act to impose a steadily increasing penalty for each overshoot of the commanded setpoint. For instance, the penalty associated with a unit altitude error grows to three times its initial value after the second overshoot. This large penalty should encourage the genetic controller to produce a smooth acquisition of target settings.

**Table 10.1:** Lateral Control Law Parameters Altered by Genetic Algorithm

| Parameter | Nominal Value | Description |
|---|---|---|
| SIEPLI | 30.0 | Heading angle deviation limit (degrees) |
| GGSI | 0.0 | Heading angle hold integral command generation gain |
| PHSINL | 20.0 | Integrated heading angle deviation contribution to commanded roll angle limit (degrees) |
| RKKSI | 2.00 | Heading angle hold proportional command generation gain |
| PHICOL | 45.0 | Indirectly generated commanded roll angle limit (degrees) |

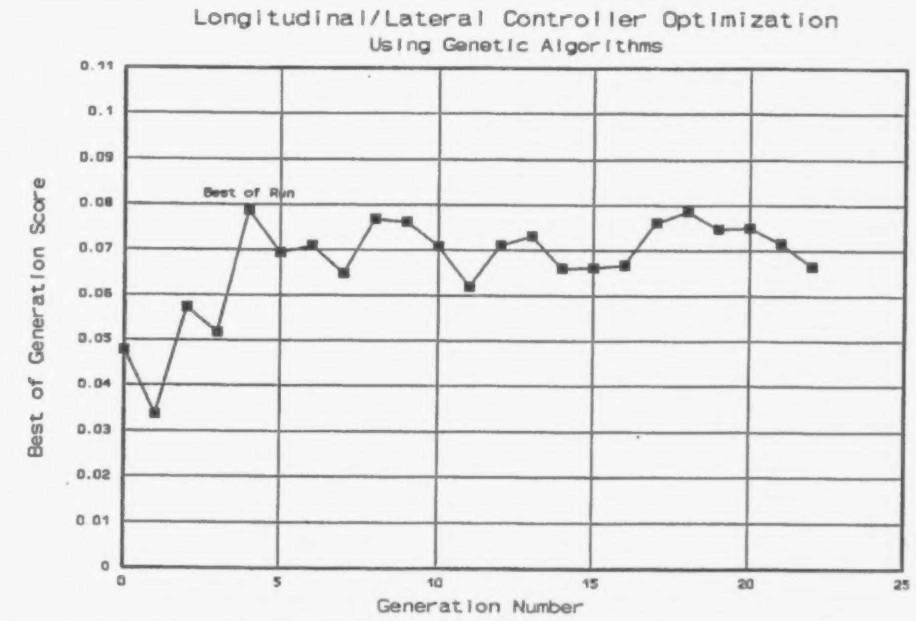

**Figure 10.1:** Best of Generation Scores for Longitudinal/Lateral Controller Development

## 10.2 Results

The genetic algorithm was executed for 22 generations, with each generation consisting of 200 individuals. Convergence results are illustrated in Figure 10.1. The best-of-run individual occurred on the fourth generation. Subsequent generations acted to improve the average performance of the population, but were unable to produce a superior individual. This trend is apparent in the curves of Figure 10.2. The figure displays the distribution of performance ratings for each individual in selected generations. Every fourth generation is graphed, beginning with the initial random population.

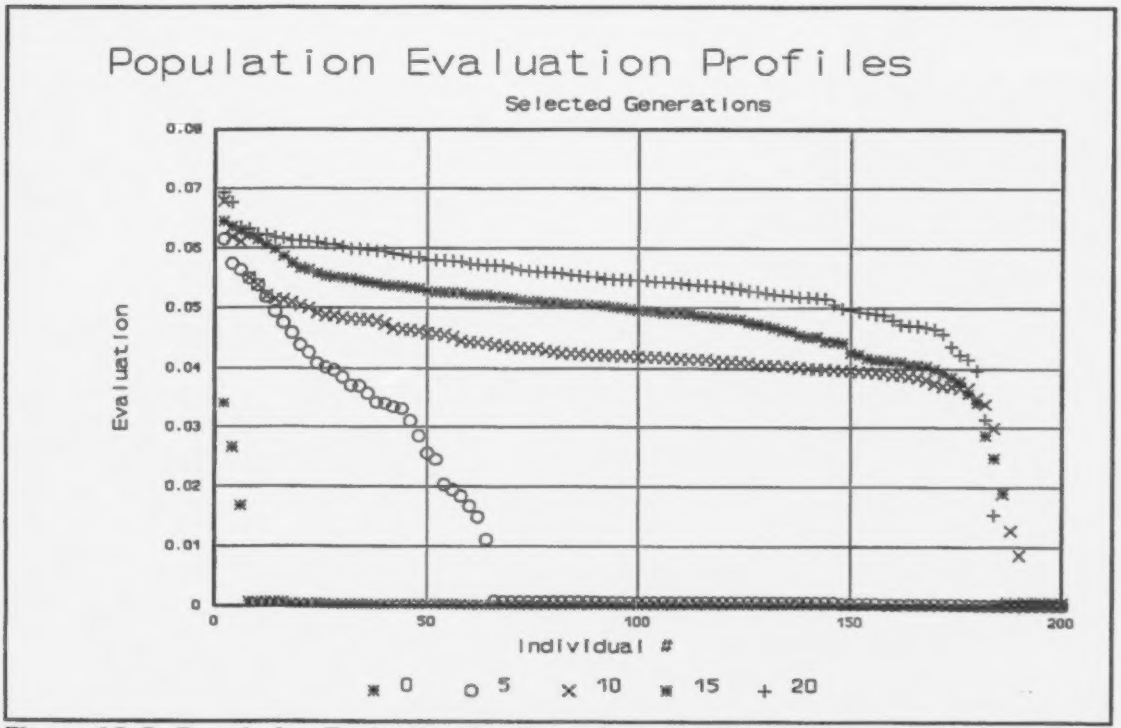

**Figure 10.2:** Population Evaluation Profiles - Selected Generations

The performance of the genetically optimized controller was evaluated through comparison with the baseline PID controller. Results were graphed for execution of the PID and genetically optimized neural network controllers over the mission profile used for G.A. training. The resulting altitude profiles are graphed in Figure 10.3 while heading profiles are depicted in Figure 10.4.

Figure 10.3 illustrates the ability of the genetically optimized controller to track commanded altitude changes with greatly reduced rise time and little overshoot. As

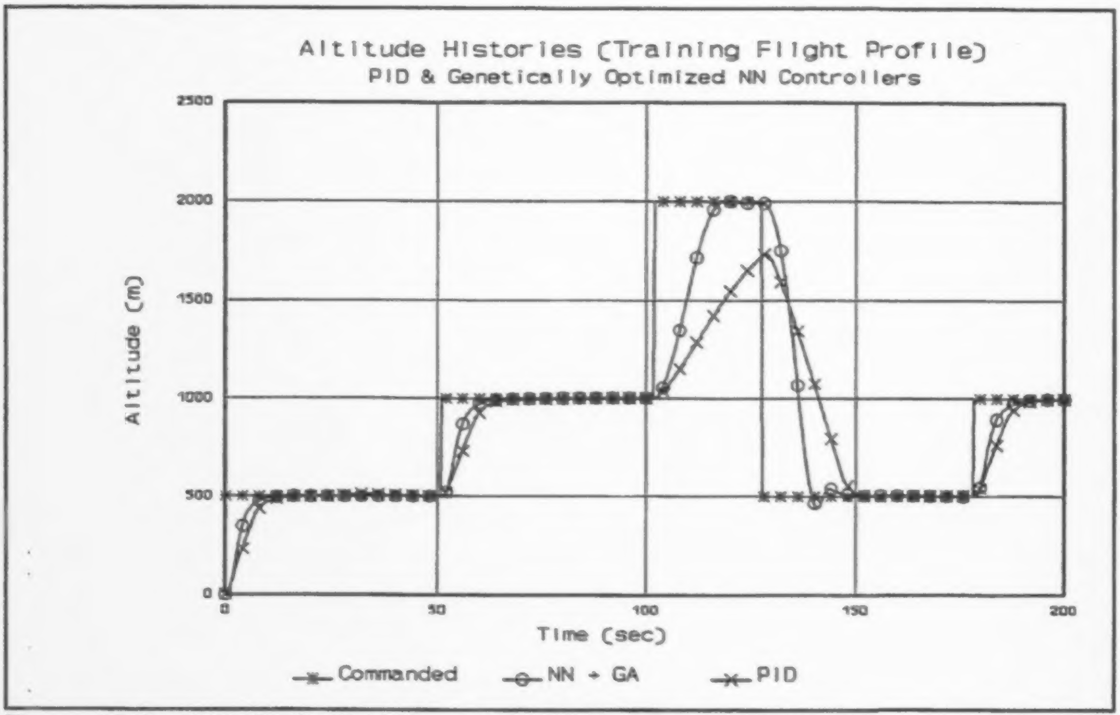

**Figure 10.3**: Genetically Optimized Altitude Profile

has been previously discussed, the apparent performance improvement is somewhat misleading. The genetic controller has been optimized for the special case of continuously available thrust and a maximum altitude transition of 1000 m. In contrast, the PID autopilot gains are appropriate to a vehicle with zero thrust vector and relatively arbitrary altitude transitions. These differences imply an important distinction in the interpretation of the results obtained. The graphs shown should be taken as an illustration of the genetic algorithm's power to specialize a general controller or other functional for use in a particular operating regime. It is not an indication that general HATT-X performance can be greatly enhanced through blind application of the genetic algorithm.

Figure 10.4 further underscores the potential difficulties in genetic algorithm application previously discussed. The heading hold control laws are active from time indices 0.0 to 75.0 and 125.0 to 155.0. The PID controller is seen to track in the direction of the commanded heading during these periods. However, the genetically optimized controller maintains an almost constant heading during the corresponding intervals. The "optimal" gain set was examined to determine the cause of this characteristic. It was discovered that the primary gain used to convert heading errors to roll angle commands had been reduced to 1.3% of its value in the baseline PID controller. As a result, heading errors produced an extremely small commanded bank

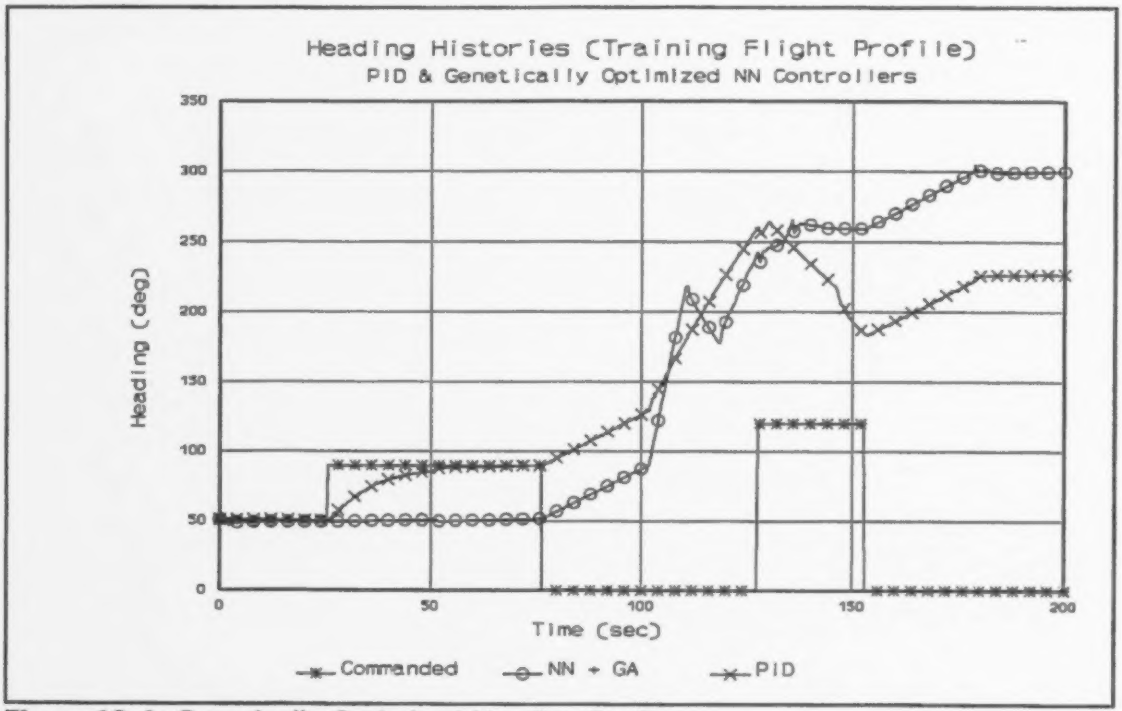

**Figure 10.4:** Genetically Optimized Heading Profile

angle, resulting in the near-constant heading profiles observed.

It is apparent that the genetic algorithm has highly optimized the controller's longitudinal loop while neglecting the performance of the lateral control laws. Two possible causes may be presented for this result. First, the genetic algorithm may not have encountered a region of gain space which allows stable operation of the lateral control laws. Second, the performance function used may have assigned insufficient weighting to lateral control performance. Either of these possibilities is a major indictment of the genetic algorithm's feasibility.

Should the first alternative hold true, it would imply a major uncertainty in the effectiveness of the genetic algorithm's search space. Twenty-two generations of 200 individuals each were executed in the course of this controller optimization. Including the original random generation, a total of 4600 possible values for the deficient gain could have been evaluated. Reproduction and crossover of similar individuals will have acted to reduce this number. Yet, sufficient alternatives should remain that it seems unlikely that such an inferior gain setting could have been selected. That such a result could be possible is a strong argument against the effectiveness of the genetic algorithm's search technique.

The second alternative presented above assumes that the magnitude of lateral penalty terms is not significant in comparision with the corresponding longitudinal terms. The genetic algorithm therefore concentrates its efforts on optimization of the longitudinal control laws. Large improvements in lateral controllability are ignored, as the relative weightings make even small longitudinal improvements appear more attractive.

This difficulty was previously encountered in development of the transport trailer controller. Significant effort was therefore expended in ensuring that the relative contributions of the lateral and longitudinal controllers to the overall evaluation would be equal. The results of this test further highlight the previously discussed difficulties inherent in determining and assessing an appropriate evaluation function.

It is noted that several literature sources (eg: Koza[7]) have provided strong arguments supporting the effectiveness of genetic algorithms in examining large search spaces. Atlantis therefore considers the second alternative (unequal weighting of lateral and longitudinal control error terms) to be a more likely explanation of the difficulties encountered.

## 11.0 Real Flight Requirements Definition

The research documented in this report has been conducted as a proof-of-concept study in preparation for eventual test flights incorporating a neural network control system with the HATT-X autonomous air vehicle developed by DRES. Several requirements for the flight system have been identified as a result of the work conducted under this contract. These requirements are identified in the following sections.

### 11.1 Algorithmic and Computational Requirements

Atlantis believes that the results of this contract have demonstrated the validity of the IDM approach to neural network control of autonomous flight vehicles. It is therefore expected that a flight vehicle implementation of a neural network controller will closely parallel the architecture used for this work.

Specifically, the recommended controller will consist of two IDM neural networks (one each for lateral and longitudinal control). The input set and network architecture for each controller will be similar to that used in this work. Network operations will cycle at the clock rate of the MPS-100 autopilot (100 Hz). A network cycle will be composed of the following operations:

a. presentation of inputs
b. calculation of output
c. evaluation of performance
d. calculation of output error
e. updating of network weights.

### 11.2 Neural Network Hardware

The HATT-X flight control system utilizes hardware and software developed by Atlantis Aerospace Corporation in consultation with the DRES Systems Integration Group. This autopilot system is termed MPS-100 and consists of several modular subsystems mounted in a common backplane. Primary computational work is conducted on a NEC V20-based processor unit (the "CPU"). A multifunction card ("MFC") based on an Intel 80196 microcontroller provides additional processor power and i/o capability for computationally intensive tasks (eg: turbojet control, inertial navigation). Finally, various I/O cards are utilized for sensor integration, servo control, and communications.

Atlantis' initial concept for incorporation of neural network control was based on integration of a second MFC with the MPS-100 autopilot. This arrangement would have allowed the use of existing, proven hardware. However, initial tests have shown that the MFC floating point throughput is inadequate for real-time execution of the neural networks.

Upon elimination of this alternative from consideration, Atlantis conducted several timing tests using a 50 MHz 486 to run optimized neural network code. These tests measured the average time required to execute one neural network cycle (consisting of input scaling, output calculation, error determination, and weight adjustment). Test results are depicted in Figure 10.5 for a variety of network sizes.

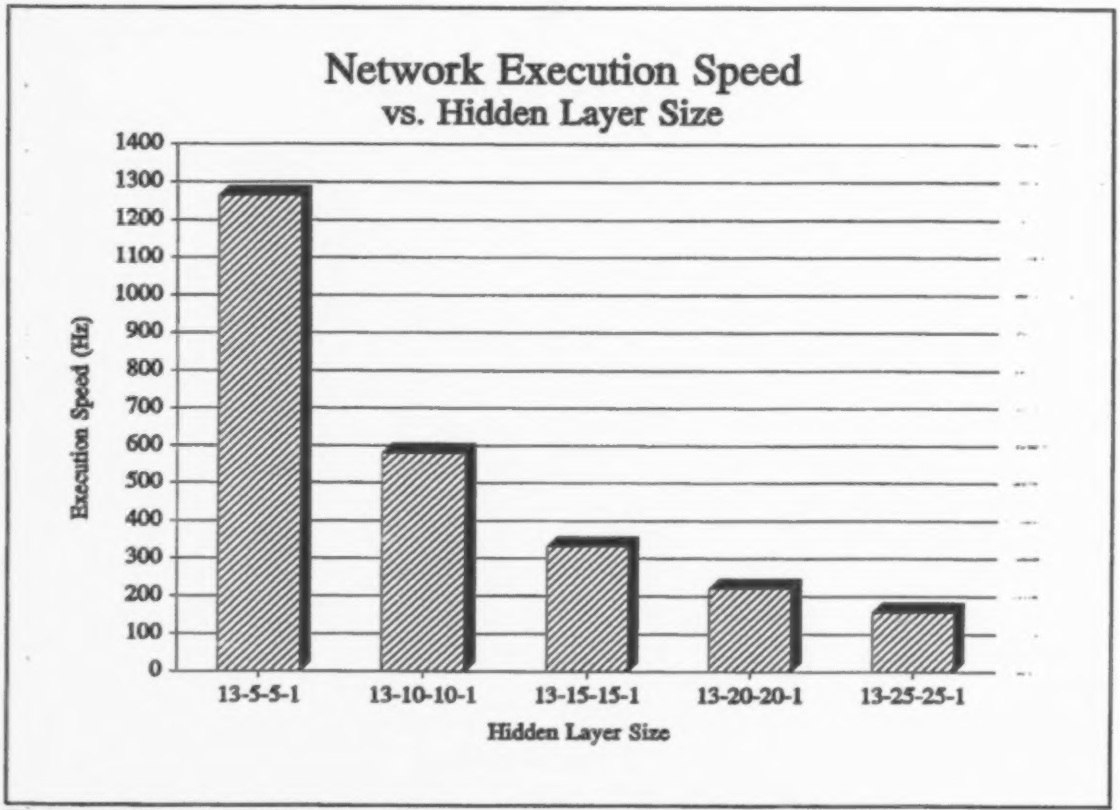

Figure 10.5: 486-50 Process Network Throughput

Note that the minimum permissible throughput is 200 Hz since the autopilot iteration rate is 100 Hz and both longitudinal and lateral networks must be cycled on each iteration. Simulation results have shown that 13-15-15-1 networks produce acceptable vehicle controllability. The test results indicate that a 486 processor running at 50 MHz will therefore meet the timing constraints. An idle margin of approximately 50 % remains for communications overhead or contingency use.

Given this result, Atlantis has recommended that the IDM networks be hosted on an industrial-grade PC. A suitable communication link will allow exchange of state and control information between this neural coprocessor and the MPS-100 autopilot.

### 11.3 Communication Interface

The communication link between the autopilot and the neural coprocessor will perform three main tasks. First, it must allow current vehicle state data to be transferred from the autopilot to the coprocessor for input to the IDM networks. Second, it must transfer the IDM network outputs to the autopilot for output to the servos. Finally, it must support status inquiries and checks as would normally occur during vehicle bench testing and prelaunch.

Data throughput requirements for the communication interface are proportional to the number of network inputs and outputs. As currently defined, each IDM network uses an identical set of thirteen inputs, and produces one output. The required data transfer rate is therefore 3000 bytes/second ((13 inputs + 2 outputs) * 100 Hz * 2 bytes per variable).

Several alternate methods were considered for implementation of the autopilot-neural coprocessor communication link. The alternatives studied were dual port ram, standard data bus, serial, and parallel.

Variable exchange via dual port ram was considered the preferred method of communication between the two processors. This approach would have allowed rapid, error-free communication of control and status information. However, no industrial grade PCs were found to provide direct support of dual port ram operations. The closest approximation available consisted of simulated dual port operations via mapping of PC memory to a VME-BUS connector.

Following this result, addition of a VME-BUS capability to the MPS-100 was considered. This alternative was quickly eliminated due to the large expenditure on hardware design and debug which would have been required.

Finally, communication via a serial or parallel interface was considered. MPS-100 currently supports serial data transfer at rates up to 9600 baud. Unfortunately, this transfer rate is inadequate to meet the 3 KByte/second requirement identified above. Atlantis expects that adequate serial transfer rates can be achieved through modification of the existing MPS-100 multifunction card or design of a serial interface card.

Implementation of a parallel data interface will also require the design of a new interface card, as all MPS-100 digital inputs are currently dedicated to turbojet operation support. Again, Atlantis feels that this option can be executed at reasonable cost.

Each system evaluated for use as a neural network host processor exceeds the required capacity for both serial and parallel communication protocols. Final selection of a serial or parallel interface is therefore not considered to be a priority at this time and has been deferred to future consideration.

### 11.4 Areas of Potential Risk

This section identifies items which pose potential risks to the success of HATT-X test flights under neural network control. Where possible, solutions or recommendations for further investigation prior to flight testing are presented.

#### 11.4.1 Observability of Required Inputs

As currently implemented, the actual canard and aileron deflections are included in the IDM network input vector. The MPS-100 autopilot does not have access to these parameters. Further simulation should be conducted to determine the importance of these variables to IDM network accuracy. Simulation results should provide a suitable course of action. Possibilities include removing the servo positions from the network input vector, predicting the current servo position in software, or system modification to permit detection of servo positions.

#### 11.4.2 Accuracy of Training Data

It is expected that the IDM networks used for test flights will primarily be trained using data obtained from SIMFAC/FLISIM. Flight performance will therefore depend on the fidelity of the FLISIM software. This is not a major concern, however, as all tests to date have demonstrated the neural network's ability to quickly compensate for errors in its initial dynamic model.

#### 11.4.3 Sensitivity to Sensor Noise

All investigations to date have been conducted using the FLISIM simulation package to exercise the IDM controller. As a result, IDM network inputs have consisted exclusively of the "correct" state variables. A major objective of the HATT-X research program is to develop methods for extracting maximum performance from low-grade sensors. During flight, these "noisy" sensor readings will be used for online training of the network inputs. It is therefore necessary to validate operation of the IDM controller in such an environment. Preliminary investigations have been conducted in this area (see Section 9.1 for details). Atlantis believes that this work should continue, and would be most effectively accomplished through addition of a representative sensor noise model to the FLISIM autopilot software package.

### 11.4.4 Sanity Checks and Override Procedures

The use of IDM neural networks is a completely novel approach to control of a flight vehicle and is largely unproven at this time. It would therefore be prudent to incorporate a number of sanity checks and overrides into the MPS-100 autopilot to ensure that risk to hardware and personnel is minimized. Current MPS-100 control laws should be executed continuously and used as a baseline against which the IDM network control actions will be evaluated. Upon triggering of certain criteria, the MPS-100 control logic should override the IDM network outputs to bring the vehicle back to a safe state. As a minimum, the evaluated criteria should include the following:

a. Decrease in vehicle altitude below a specified minimum safe height.
b. Transition of roll or pitch angles and their rates beyond specified limits.
c. Large or sustained difference between PID and IDM control outputs.
d. Onset of IDM network instability.

### 11.5 Flight Test Plan

Atlantis recommends an incremental approach to the implementation of neural network control on the HATT-X vehicle. Addition of a small, clearly defined capability to each flight will allow the evaluation of each component's effect on vehicle performance.

It is anticipated that flight testing will progress through the following stages. Each stage may consist of one or more individual flights depending on progress achievement and modifications required. Flights will be conducted using either the manned (Seneca) or unmanned (HATT-X) research aircraft as appropriate.

| | |
|---|---|
| Seneca 1 | Neural network control of longitudinal flight. No on-line learning. This flight allows validation of the basic mapping embedded in the IDM network. |
| Seneca 2 | Neural network control of longitudinal flight. On-line learning enabled. This flight allows validation of on-line learning techniques in the face of noisy sensor inputs. |
| Seneca 3 | Neural network control of combined longitudinal/lateral flight. This flight extends the results of Seneca 2 to full aircraft control. |
| HATT-X 1 | Neural network control of longitudinal flight. PID control of lateral flight. This configuration will allow assessment of the neural controller's altitude tracking capabilities. Use of the lateral PID control laws guarantee control of the vehicle's ground track, facilitating range safety. |

| | |
|---|---|
| HATT-X 2 | PID control of longitudinal flight. Neural network control of lateral flight. Roll angle hold is used exclusively to allow assessment of lateral tracking and stability. |
| HATT-X 3 | Neural network control of longitudinal and lateral flight. Addition of lateral heading hold waypoints. Allows evaluation of lateral/longitudinal combined controllability. |

## 12.0 Recommendations for Future Work

Several avenues for future research have been identified as a result of this work. Atlantis believes that two potential research directions hold great promise. First of these is hardware implementation of the IDM control network for testing in actual vehicle control. This may be considered the "applied development" branch. A second development path involves further simulation, analysis, and development of intelligent control techniques. Atlantis has termed this the "pure research" branch.

The applied development branch entails integration of a neural network with the MPS-100 autopilot for actual vehicle control. Atlantis has examined the feasibility of this goal and believes it to be readily attainable. Necessary requirements and a potential development approach to such a project have been documented in Section 11 of this report. Completion of the applied development branch will entail the following tasks:

a. Optimization of the existing neural network software for execution in a real-time environment.
b. Development of a suitable communication interface between the MPS-100 autopilot and the neural network host processor.
c. Development of MPS-100 software in support of neural network operations.
d. Development of an appropriate simulation model for the DRES manned research aircraft (Seneca) dynamics.
e. IDM network training on the developed Seneca simulation model.
f. IDM network training on the existing HATT-X simulation software (FLISIM)
g. Interfacing of the MPS-100/Neural Network assembly to the Seneca control system.
h. Test flights (Seneca and HATT-X), data collection, and documentation as appropriate.

The pure research branch involves further research and simulation in support of applied development. Potential topics for investigation include the following:

a. IDM network stability analysis in noisy environments.
b. Alternative IDM network topologies (eg: radial basis function or recurrent rather than back-propagation neural networks.
c. Intelligent methods for outer loop optimization.

Of these alternatives, Atlantis believes item c. to be the most promising. Genetic algorithms have been shown to possess several characteristics which make their application to controller design difficult. However, the results obtained illustrate that potential performance improvements can be obtained by applying a suitable optimization technique to calculation of controller settings. A recent literature search has identified several abstracts related to on-line adaptation of PID controller gains via state feedback. Neural networks or fuzzy logic are identified as the intelligent agents used in controller modification. Little is currently known about the techniques involved in this work. Atlantis strongly recommends therefore that an investigation into this area be conducted in any future contract.

## 13.0 List of Deliverables

The following items were purchased in the course of this contract. These items currently remain on loan to Atlantis Aerospace in anticipation of their use in a follow-on contract.

| **Description** | **Serial Number (if available)** |
|---|---|
| Microway 486-50 workstation | MB24100387 |
| Ancer 14 " VGA color monitor | G/H-20600534 |
| 213 MB SCSI hard drive | 1024248-4 |
| 340 MB SCSI hard drive | 1028045-5 |
| Perfect Electronics 486-66 workstation | 4648 |
| NEC MultiSync 4FGe color monitor | 3X17150KA |
| Microway Numbersmasher-860 32 MB | SN 1506 |
| Microway Numbersmasher-860 32 MB | SN 1582 |
| Ziatech Star system (dual processor) | 00435 |
| Microway i860 C/C++ compiler | 121423 |
| Microway i860 Fortran compiler | 121423 |
| Microway i860 Pascal compiler | 123465 |
| Vast 860 code vectorizer | 121423 |
| PPS-860 pipeline scheduler | 121423 |
| IMSL libraries for i860 | 121423 |
| DOS 5.0 (2 copies) | |

## 14.0 References

1. Kong, Seong-Gon, and Kosko, Bart, "Adaptive Fuzzy Systems for Backing up a Truck-and-Trailer," in *IEEE Transactions on Neural Networks*, vol. III, March 1992, pp. 211-223

2. Nguyen, D., and Widrow, B., "The truck backer-upper: An example of self-learning in neural networks," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN-89)*, vol. II, June 1989, pp. 357-363

3. Atlantis Aerospace Corporation, "Investigation of Simulated Flight Control Using An Inverse Dynamic Model Neural Network - Interim Report", Atlantis Aerospace Corporation, Brampton, 1993

4. Atlantis Aerospace Corporation, "Investigation of a Simulated Neural Network Autopilot for Altitude Control - Final Report", Atlantis Aerospace Corporation, Brampton, 1992

5. Langley, A.M., "Practical Limitations on the Effective Use of Adaptive Neural Networks for Solution of the Dynamic Programming Model", Undergraduate Thesis, Division of Engineering Science, University of Toronto, 1992

6. Markov, A.B., "ROBOT-X Software Specifications (U)", DRES RN 429, Medicine Hat, 1984

7. Koza, John R., "Genetic Programming: On the Programming of Computers by Means of Natural Selection", MIT Press, Cambridge, 1992